'68' MICRO JOURNAL

$2.95 USA

Australia	A $ 4.75	New Zealand	NZ $ 6.50
Singapore	S $ 9.45	Hong Kong	H $23.50
Malaysia	M $ 9.45	Sweden	30:-SEK

VOLUME VI ISSUE V • Devoted to the 68XX User • May 1984

"Small Computers Doing Big Things"

0 74470 12810 05

WE DON'T PLAY GAMES

X-12+ A SERIOUS COMPUTER IN A DESKTOP PACKAGE

Multiprocessor Technology - Combination of 8, 16 and 32 bit types

1.0 Megabyte Memory - Insures no limitation on programs

"Winchester" Disk System - Fast response, large storage capacity

UniFlex* Operating System - The standard of comparison

Hardware Floating Point - Unmatched speed in a small system

Up to Three Terminals - Instant expansion

*Trademark of Technical Systems Consultants

SWTPC

SOUTHWEST TECHNICAL PRODUCTS CORPORATION
219 W. RHAPSODY
SAN ANTONIO, TEXAS 78216 (512) 344-0241

'68' MICRO JOURNAL

Portions of the text for 68 MICRO JOURNAL was prepared using the following furnished hard/software.

COMPUTERS-HARDWARE
Southwest Technical Products
219 W. Rhapsody
San Antonio, Texas 78216
S09-5/8 DMF disk-CDS1-8212W-Sprint 3 Printer

GIMIX Inc.
1337 West 37th Place
Chicago, IL 60609
Super Mainframe-OS9-FLEX-Assorted Hardware

EDITORS-WORD PROCESSORS
Technical Systems Consultants, Inc.
111 Providence Road
Chapel Hill, NC 27514
FLEX-Editor-Processor

Great Plains Computer Company, Inc.
PO Box 916
Idaho Falls, ID 83401
STYLO-Mail Merge

CONTENTS

Send All Correspondence To:

Computer Publishing Center
68 MICRO JOURNAL
5900 Cassandra Smith
PO Box 849
Hixson, TN 37343
615 842-4600

68' Micro Journal is published 12 times a year by Computer Publishing Inc. Second Class Postage Paid ISSN 0194-5025 at Hixson, Tenn. and additional entries. Postmaster: send Form 3579 to 68' Micro Journal, PO Box 849, Hixson, Tennessee.

SUBSCRIPTION RATES

USA

1-Year $24.50 2-Years $42.50 3-Years $64.50

FOREIGN

See Page 60

Items Submitted for Publication

Articles submitted for publication should be accompanied by the authors **full name, address, date and telephone number.** It is preferred that articles be submitted on either 5 or 8 inch diskette in TSC Editor format or STYLO format. All diskettes will be returned.

The following TSC Text Processor commands **ONLY** should be used (due to our proportional processor): .sp space, .pp paragraph, .fi fill and .nf no fill. Also please do not format within the text with multiple spaces. The rest we will enter at time of editing.

STYLO commands are all acceptable except the .pg page command, we print edited text files in continous text.

All articles submitted on diskettes should be in TSC FLEX™ format, either FLEX2 6800, or FLEX9 6809 any version.

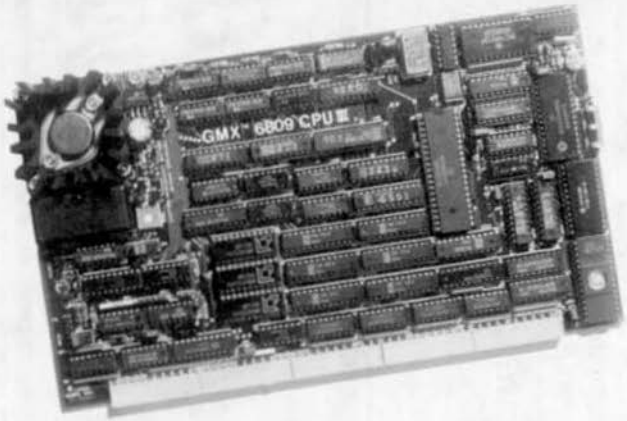
If articles are submitted on paper they should be on white 8X11 bond or better grade paper. No hand written articles (hand written or drawn art accepted). All paper submitted articles will be photo reproduced. This requires that they be typed or produced with a dark ribbon (no blue), single spaced and type font no smaller than 'elite' or 12 pitch. Typed text should be approximately 7 inches wide (will be reduced to column width of 3 1/2 inches). **Please use a dark ribbon!**

All letters to the editor should also comply with the above and bear a signature. Letters of 'gripes' as well as 'praise' are solicited. We attempt to publish all letters to the editor verbatim, however, we reserve the right to reject any submission for lack of 'good taste'. We reserve the right to define what constitutes 'good taste'.

Advertising: Commercial advertisers please contact the 68 Micro Journal advertising department for current rate sheet and requirements.

Classified: All classified must be non-commercial. Maximum 20 words per classified ad. Those consisting of more than 20 words should be figured at .35 cents per word. 20 words or less $7.50 minimum, one time, paid in advance. No classified ads accepted by telephone.

GIMIX STATE OF THE ART 6809 SYSTEMS FOR THE SERIOUS USER.



GIMIX has 19MB or high performance 47MB Winchester Drive Systems and/or Floppy Disk Drive Systems.

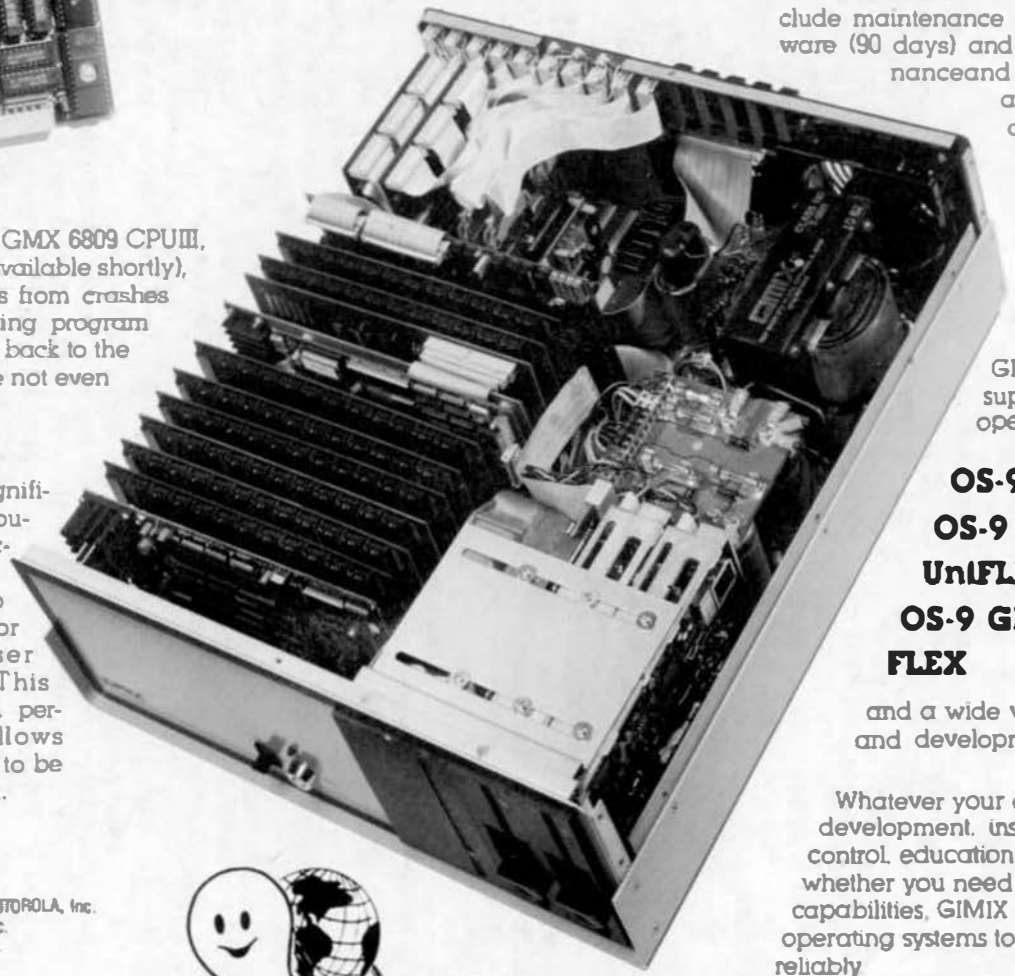
For the user who appreciates the need for a bus structured system using STATIC RAM and powered by a ferro resonant constant voltage transformer.

GIMIX has single user systems that can run both FLEX and OS-9 or Multi user systems for use with UniFLEX or OS-9.

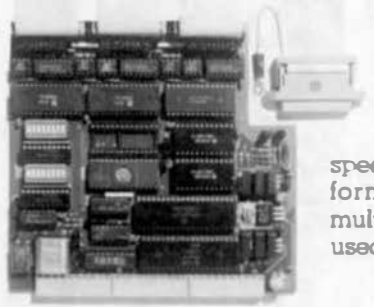
GIMIX versions of OS9 and UniFLEX include maintenance and support by Microware (90 days) and TSC (1 year). Maintenanceand support after this period are available at extra cost.

(NOTE: this support and maintenance is only for use with approved GIMIX hardware)

For the ultimate in performance, the Unique GMX 6809 CPUIII, using either OS-9-GMXIII or UniFLEX GMXIII (available shortly), gives protection to the system and other users from crashes caused by defective user programs, e.g. During program development, a programmer who crashes goes back to the shell or the debugger, while the other users are not even aware anything occurred.



The intelligent serial I/O processor boards significantly reduce system overhead by handling routine I/O functions, thereby freeing up the host CPU for running user programs. This speeds up system performance and allows multiple terminals to be used at 19.2K baud.



GIMIX 6809 systems support five predominant operating systems:

OS-9 GMX III,
OS-9 GMX II,
UniFLEX,
OS-9 GMX I,
FLEX

and a wide variety of languages and development software.

Whatever your application: software development, instrumentation, process control, educational, scientific or business; whether you need single or multi-user capabilities, GIMIX has hardware and the operating systems to get the job done reliably.

Please phone or write if you need further information.

GIMIX INC.
1337 WEST 37th PLACE • CHICAGO, ILLINOIS 60609 • (312) 927-5510 • TWX 910-221-4055

Microware presents 4 new OS-9 software packages.

1 LEVEL II PRINT SPOOLING SYSTEM

This versatile package gives your OS-9 Level Two System a complete print spooling management capability for time-sharing applications. Features of the spooling system are:

- *Handles up to seven independent spooling devices and queues with "print on first available device" feature.*
- *Prints large block header pages between listings with date, time, user name and job name.*
- *Multiple listing copy option.*
- *Complete forms change capability for each job and device.*
- *Prints formatted or unformatted listings.*
- *Status command displays print queues and status.*
- *User can kill or change priority of queued jobs.*

*Available **only** for OS-9 Level Two Systems.*

Suggested List Price: $150.00 Manual Only: $15.00

2 RMA RELOCATABLE MACRO ASSEMBLER

At last — a full feature relocatable macro assembler and linkage editor for OS-9. RMA permits sections of assembly language programs to be independently assembled to "relocatable object files". The linkage editor takes any number of program sections and/or library sections and combines them into a single executable OS-9 memory module. Global data (including indexed and direct addressing modes) and program references are automatically resolved in the process. The macro facility permits commonly used statement sequences to be defined, then used within the program with appropriate parameter substitution. RMA also supports conditional assembly and library source files.

Suggested List Price: $200.00 Manual Only: $20.00

3 OS-9 FILE HANDLER TOOLBOX

Introducing a special toolbox for OS-9 users who do a lot of file manipulation! A collection of 12 useful OS-9 command programs. Most can be used as "filters" using OS-9 pipeline facilities. Included are:

D *— unformatted directory listing with "wild card" matching*
Compress *— does character compression on text files.*
Expand *— restores a "compressed" file to the original state.*
Split *— breaks a file into smaller files.*
Space *— indents lines with optional spacing between lines.*
Code *— decodes any key on a keyboard to hex.*
Qsort *— quick sort for small files, directories, etc.*
Pr *— versatile formatted file printing utility.*
Tr *— transliterates text pattern to substitution pattern.*
Grep *— searches file for a pattern and prints matching lines.*
Xmode *— same "tmode" except changes are made to the device descriptor.*
Count *— counts words, lines, or characters within a text file.*

Suggested List Price $85.00

4 ENTERTAINMENT PACK I

A collection of games and other interesting programs that are not only entertaining but serve as good instructional examples of Basic09 programming techniques. All programs include complete Basic09 source files and can be easily edited to run on standard alphanumeric or graphics terminals.

Blkjak *— A Vegas-rules blackjack game.*
Clk *— graphical display of a wall clock on your terminal.*
Dogs *— Greyhound racing with simulated graphics.*
Eliza *— Basic09 version of the famous artificial intelligence simulation of natural language dialogue with a psychiatrist.*
Haiku *— Program that creates original "haiku" prose.*
Quest *— a mini-"Adventure" game.*
Rats *— find your way out of a computer-generated maze — from a rat's point of view.*
Towers *— a graphical display of the solution to the "Tower of Hanoi" puzzle.*

Suggested List Price: $85.00

MICROWARE®

Microware Systems Corporation
P.O. Box 4865 • Des Moines, IA 50304
515-279-8844 • Telex 910-520-2535

OS-9 and Basic09 are trademarks of Microware and Motorola. UNIX is a trademark of Bell Labs

"More help than any other thing I have ever read, except 68 Micro Journal, of course. Come on with the C book on Norm Commo's column."

FLEX™ USER NOTES
THE 6800-6809 BOOK

By: Ronald W. Anderson
As published in 68 MICRO JOURNAL™

"SWELL! - Best buy EVER in a 6800/6809 book. JTW"

The publishers of 68 MICRO JOURNAL are proud to announce the publication of Ron Anderson's **FLEX USER NOTES,** in book form. This popular monthly column has been a regular feature in 68 MICRO JOURNAL SINCE 1979. It has earned the respect of thousands of 68 MICRO JOURNAL readers over the years. In fact, Ron's column has been described as the 'Bible' for 68XX users, by some of the world's leading microprocessor professionals. Now all his columns are being published, in whole, as the most needed and popular 68XX book available. Over the years Ron's column has been one of the most popular in 68 MICRO JOURNAL. And of course 68 MICRO JOURNAL is the most popular 68XX magazine published.

As a SPECIAL BONUS all the source listing in the book will be available on disk for the low price of: FLEX™ format only — 5" $12.95 — 8" $16.95 plus $2.50 shipping and handling, if ordered with the book. If ordered separately the price of the disks will be: 5" $17.95 — 8" $19.95 plus $2.50 shipping and handling.

Listed below are a few of the **TEXT** files included in the book and on diskette.

All TEXT files in the book are on the disks.

LOGO.C1	File load program to offset memory — ASM PIC
MEMOVE.C1	Memory move program — ASM PIC
DUMP.C1	Printer dump program — uses LOGO — ASM PIC
SUBTEST.C1	Simulation of 6800 code to 6809, show differences — ASM
TERMEM.C2	Modem input to disk (or other port input to disk) — ASM
M.C2	Output a file to modem (or another port) — ASM
PRINT.C3	Parallel (enhanced) printer driver — ASM
MODEM.C2	TTL output to CRT and modem (or other port) — ASM
SCIPKG.C1	Scientific math routines — PASCAL
U.C4	Mini-monitor, disk resident, many useful functions — ASM
PRINT.C4	Parallel printer driver, without PFLAG — ASM
SET.C5	Set printer modes — ASM
SETBAS1.C5	Set printer modes — A-BASIC
	(And many more)

**Over 30 TEXT files included in ASM (assembler) — PASCAL — PIC (position independent code) TSC BASIC-C, etc.

NOTE: .C1, .C2, etc. = Chapter 1, Chapter 2, etc.

This will be a limited run and we cannot guarantee that supplies will last long. Order now for early delivery.

"Finally I don't have to type in all that source code, a GREAT idea. Thanks and good luck, you guys are the best! TOA"

"I will keep this (FLEX USER NOTES) book always beside my computer. I have not ever, with any magazine, received as much information as this one book, and 68 MJ. DIB"

Foreign Orders Add $4.50 S/H

Softcover — Large Format

Book only: **$7.95** + $2.50 S/H

With disk: 5" **$20.90** + $2.50 S/H

With disk: 8" **$22.90** + $2.50 S/H

See your local S50 dealer/bookstore or order direct from:

Computer Publishing Inc.
5900 Cassandra Smith Rd.
Hixson, TN 37343
(615) 842-4601

™FLEX is a trademark of Technical Systems Consultants

!!! FREE !!!

Published Monthly by Computer Publishing Inc., Hixson, TN.

$1.95

Bulk Rate
U.S. Postage
PAID
Chattanooga, TN
Permit No. 357

Color Micro Journal

The Color Computer Monthly Magazine

$1.95 per issue Vol. 1, Issue 2 October, 1983

THIS 'N THAT

The BIG NEWS this month is that OS-9 has finally arrived for the Color Computer. The ASTOUNDING part of the Radio Shack OS-9 Package, besides the price, is the DOCUMENTATION. You 'Old Time Radio Shack Followers' will not believe what you see. Jon Shirley has been telling us that the main reason for the "lack" of documentation with a lot of their products was the restrictions placed on releasing that information by Microsoft; I

OS-9 on the COLOR COMPUTER

One of the "Operating Systems of the Future" is now available for the "little old Color Computer"; OS-9. Freely translated, OS-9 means "Operating System for the 6809" (OS-9 is now being written for the 68000, also). Since it is fairly obvious that UNIX and "UNIX-Type" Operating Systems will be running on just about every computer to come out in the next few years, a whole new language is beginning to appear on the horizon.

Color Computer OS-9; the Package

We had been running a preliminary release of OS-9 on the Color Computer for a few weeks, and received the "official Radio Shack" version for Review a couple of days ago. To put it mildly, this package is IMPRESSIVE! For $69.95 (Radio Shack Catalog Number 26-3038), you receive a 9 1/2" x 7 5/8" x 2" package containing 4

TMP (Total Management Planning) OS-9 SOFTWARE

THESE OFF-THE-SHELF APPLICATION PACKAGES HAVE CHANGED THE PICTURE FOR PROFESSIONAL SOFTWARE USERS

Here are the comprehensively supported OS-9 application software packages that discerning professionals are selecting today. Each one runs stand-alone or in combination with other TMP packages.

TMP/CALC is a *new-generation* "Electronic Spreadsheet" package that goes a quantum leap beyond ordinary Calc software. Superior speed, and extra features like Dynamic Calculations and Dynamic Overlay, mean you can change a cell *anywhere* on the spreadsheet and get automatic updates wherever they belong. No more "wrap-around" since output to the printer is formatted, "Help" screens prompt without erasing data, and the rows and columns on any worksheet are limited *only* by available memory.
ONLY $250

TMP/Manager is a structured database manager, and is the ultimate driving force behind any total *Management/Marketing Planning* system. Its built-in select/sort/merge module — and optimized machine code — makes it superfast, and provides unusually nimble data manipulation characteristics. Other outstanding features are detailed prompting, easy identification of data fields, and an excellent security system.
ONLY $500

TMP FreeForm fills the void between a structured DBMS and word-processing; it's an "electronic index card" package with an endless list of valuable applications such as parts cataloging, look-up inventories and listings of *any* kind. Type anything on up to 32,000 electronic "cards" of up to 9 pages each that reside in "file drawers" (databases). Up to 13 key-words can call each page.
ONLY $150

TMP/Front-End integrates any or all of the packages described above, and integrates TMP with word-processors, BASIC and other high-level languages.
NO CHARGE with TMP/Manager

TMP T.M.

TMP PACKAGES ARE EXCEPTIONALLY FRIENDLY. EACH COMES WITH MULTI-MEDIA TRAINING AND OUTSTANDING DOCUMENTATION

AUDIO CASSETTE TRAINING TAPES are provided with each TMP software package. These crisp, professionally produced training aids take a new user through "hands-on" exercises that instantly build confidence, and minimize reference to manuals.

CLEAR DOCUMENTATION is specially prepared for each TMP package. Remarkably readable because they are conversational in style and logically organized, this series of manuals *further* assures quick user comfort and productivity with TMP software. Includes sections on how to best utilize the system; integrating user-written software with TMP packages, and other bonus topics.

VIDEOTAPE PROGRAMS are available to TMP Dealers. They dramatically take the viewer into real business environments for a close look at how a TMP software package is utilized. Video programs can be personalized for specific organizations. Programs may be specified for most popular tape formats.

THE BOTTOM-LINE IS THIS: Software is only as effective as its training and documentation. TMP packages are *incomparable* in both areas.

RUN TMP IN A SMOKE SIGNAL CHIEFTAIN™ COMPUTER TO EXPERIENCE THE KIND OF SPEED AND POWER YOU CAN EASILY AFFORD *TODAY*

Endurance-Certified Chieftains are consistently among the fastest computers in CPU and I/O speeds according to the widely respected BENCHMARK REPORTS. That, combined with almost legendary reliability, makes Chieftains the logical choice in computers using OS-9 operating systems.

Configurations range from floppy-disk systems to multi-user, multi-tasking Winchester hard disk systems with tape back-up; performance approaching mainframes at prices you can live with.

TMP software packages for most OS-9 systems and Chieftain computers are offered exclusively from Smoke Signal. Inquiries from non-Smoke Signal dealers are invited. Call (213) 889-9340.

SMOKE SIGNAL
Chieftain Computers
31336 Via Colinas • Westlake Village, CA 91362
(213) 889-9340

Flex User Notes

Ronald W. Anderson
3540 Sturbridge Court
Ann Arbor, MI 48105

More Assembler Vs. Compiler

Dan Farnsworth received a copy of last month's column at the same time as it was sent to '68' or possibly a little sooner, and he has responded quickly.

"First, a few remarks on your last response. I do not agree that your loop and decision instructions are any clearer than mine in assembler. They both have approximately the same number of characters and the meaning is perfectly clear. Your program example using the 16 bit integer arithmetic only proves that you didn't use the right math package. You admit to having a Floating Point routine. Why didn't you 'INCLUDE FLOAT.LIB', and delete all the functions that weren't needed? Let me put the shoe on the other foot, if you had written your program in PL/9 with 7 digit accuracy and then found out you needed 10 digit accuracy, what would you do? In my Floating Point routine all you would have to do is EQU PC (precision counter) to 10 and reassemble the program.

"Your reference to a listing of 25 pages compared to 6 or 7 pages of HLL code is way off target. While I admit you would create a 25 or 30 page listing, 20 of those pages would be from your library of previously written and debugged programs, leaving only 6 or 7 pages to input. Let us not compare primative assembly language techniques to HLL's when I am advocating a much more sophisticated approach.

"The power of any programming system is in the commands that are available. I have 350 commands in my runtime package and another 20 routines that I can call from disk. Compare this with the 30 to 50 commands available in HLLs. There are 15 different ways I can output data to the screen or printer. Believe me, that is pure computer power at your fingertips.

"The mathematics package is the big stumbling block for assembly language programming. Once this problem is solved the road is down hill all the way. I will share with your readers the 10 digit BCD package I use for the Business programs. BCD was chosen because there are no conversion errors that you get when going from binary to decimal. Five bytes are required to contain a ten digit BCD number and the range is from -20 million to +80 million. Four 5 byte work areas are required and these are named NUMBER, TOTAL, BUFFER & MULBUF. These are sequential so their relationship to each other is fixed. The source code for the add and subtract routines is as follows:

```
NUMBER RMB 5
TOTAL RMB 5
BUFFER RMB 5
MULBUF RMB 5

* SUBR TO ADD BUFFER TO TOTAL
* WITH RESULTS IN TOTAL

ADDBCD CLC             CLEAR THE CARRY BIT
       LDB #5          NUMBER OF BYTES TO ADD
       LDX #TOTAL+5    POINT TO LSB+1
ADDBCI LDA 0,-X        DECR X AND GET BYTE
       ADCA 5,X        ADD  BUF BYTE TO TOTAL BYTE
       DAA             DECIMAL ADJUST
       STA 0,X         SAVE RESULT IN TOTAL
       DECB            DECR THE BYTE COUNTER
       BNE ADDBCI      LOOP UNTIL DONE
       RTS

* SUBR TO SUBTRACT BUFFER FROM TOTAL

SUBBCD LDX #BUFFER     POINT TO SUBTRAHEND
       JSR COMP        COMPLEMENT BCD NUMBER
       BRA ADDBCD      GO ADD COMPLEMENTED NBR
```

"In the next installment I will present the source code for GETNUM, the number input routine. In the next few months the complete BCD package will be presented. You Monday morning quarterbacks are invited to make any constructive comments. I'm an old Dog but I would like to learn some new tricks."

Response

I'm going to make a general response and then go with some specific examples that might make a point or two for me. First, of course the loop and decision instructions aren't any longer in Assembler than in PL/9 or "C". However the examples Dan gave lumped the "procedure" into one line marked (procedure). Certainly I don't believe, and neither does Dan that his procedure in assembler will be as short as my procedure in PL/9 or "C" or Pascal. The examples below will prove my point. While I do agree that the loop control statements are no longer in assembler, I'm absolutely not willing to agree that they are just as clear if only because there is so much code between the start end end of many loops. Of course a very good programmer like Dan would break a long section of code into short subroutines and the code in the loop could be only a few lines containing subroutine calls, but few of us have enough discipline to do that.

Dan made reference to my example of the machine program in assembler, and asked why I didn't use my floating point package. Perhaps the example was not the best. I had used 16 bit integer arithmetic for some hardware limitations. At the time, I had to squeeze the program into one 2K ROM, and the floating point package was 1.5K long. It was more an exercise in "let's see if we can do that program in 2K and get enough accuracy for the application." We did, but it was too difficult to maintain as the product concept changed the program requirements. I'm sorry I didn't have absolutely equivalent programs to compare. I still don't in this reply, because the two floating point packages I have do not have the same capabilities.

I must admit that Dan has a point about the length of program listing. I didn't take into account the "library functions" in the assembler program version as I did in the compiled version. If you are not confident in your math package enough so that you can "no list" it, you shouldn't be using the math package for any serious application anyway. I still don't believe it will be as short as in my HLL, particularly for "number crunching" applications.

You have 350 commands available and actually prefer that to my 30 or 50????? I must be missing something. How can 350 arbitrary and very "particularized" commands be better than 50 general ones? Do you have a 7 page EQU library file that you include in every assembler program you write, so you can use those subroutines?

The whole 6809 instruction set is only about 60 instructions if you consider the regularity and don't count every variation on a theme. (I know someone is going to say that the 6809 has 1423.786923 instructions). That is, LDA 0,X LDA 0,Y

LDA 0,U LDA 0,S LDB 0,X... etc. You could make 8 or 12 instructions out of those or you could generalize LD(R) n,(R) and get a bunch of instructions down to one general one.

You say you have 15 ways to output data to the screen! You probably do it with 15 different subroutine calls. YUCH and PHOOEY! You miss the point of simplicity in the HLLS if you think that is better!!!!

```
PRINTR(PI,6,1);
PRINTR(PI,12,6);
PRINTR(PI,20,4);
```

Result in my PL/9:

```
3.1
 3.141593
                3.1416
```

Most of the assembler programmers I know would write three subroutines and call them PR061, PR126, and PR204 respectively. The value is in generalization of instructions, not number of instructions! One instruction can give me more than 15 ways to output a number to the screen if it is general enough, and I have only to remember that one single instruction (or construction if you like).

Your BCD math package looks nice. Based on what you said, I expect that you have fixed the decimal point two digits from the right end of the number, and that your integer arithmetic is therefore in pennies. Your stated limit of +80,000,000 is 8 digits, so I think it is safe to assume that your maximum number is actually +80,000,000.00.

I'm sure you will tell us later how you handle monthly interest calculations, say of 14.5% annual interest, which make the monthly rate 1.2083333, or the multiplier for the interest 0.01208333? I am not interested in a fixed point arithmetic for what I need to do in general. Fixed point results in the problems I mentioned last month of overflow or underflow in my programs unless I carefully monitor each calculation for proper scaling. If I need calculation accuracy to six places after the decimal point, I will need to carry out the calculations to 15 or 18 digits if I use fixed point. That means that I have to calculate 30 to 36 digit intermediate results for multiply.

I am not saying that I couldn't use fixed point, just that it is a pain, and after all, the computer is supposed to worry about things like that for me so I don't have to, isn't it?

I remind you and our readers that you have not shown the code for your COMP routine. The 10's complement of a ten digit number takes a little time to perform. Also you have to get your value into the register you call NUMBER. That requires a 5 byte move routine.

Compare Assembler and Compiler

I said last time that I would let Dan have the last word, but he has challenged me to prove something or show how compiler is easier to use than assembler. I will therefore present some program listings and indicate the relative difficulty of using assembler programming for floating point calculations. For a simple example, I am going to program a subroutine to calculate the weight of a steel cylinder given its radius and height.

```
WEIGHT = PI * RADIUS * RADIUS * RHO * HEIGHT
```

RHO is the density of steel. Since the ASCII character set doesn't include greek letters, I have to spell it out. If radius and height are in inches, and the weight is to be in ounces, then RHO is the density in ounces per cubic inch, approximately 4.528.

I have two math packages with which to demonstrate. The first is TSC's BCD floating point package. I'm making the assumption that a 6809 version is available, which is valid, since the 6800 version could be reassembled to run on a 6809, though the code wouldn't be optimized. This package has defined some "floating point" registers. In general a register is defined as:

```
XSIGN RMB       $00=+  $FF=-
XOP RMB 4       first digit always 0
XEX RMB 1       binary signed exponent
```

The package has provision for variable precision, but this gives 7 digits, comparable to the other two packages. The three floating point registers that we need to use are X, Y, and R. X and Y generally contain the two arguments for an operation, and the result appears in the R register. Note that I am not talking about the 6809 machine registers, but about some software defined "floating point" registers. We are going to write a simple program module to perform the calculation. In each of the three examples we need to define the constants PI and RHO, and then to reserve space for the varibles HEIGHT, RADIUS, and WEIGHT. I've assumed that HEIGHT and RADIUS somehow magically got put into the varibles of those names, for all three.

Listing WTBCD is the code to perform the calculation with the BCD package. Note that most of the code, (at least the majority of it) is taken up with moving arguments into the X and Y registers, and moving the result from the R register.

I have written a binary math package that uses the User stack for calculations. Part of that package is a simple math interpreter that reads a list of argument addresses and operation codes and performs the math. The code for the calculation is considerably reduced at the expense of increasing the complexity of the math package. Listing WTBIN is the code for this package. JSR MATH causes the math interpreter to scan the list of arguments and operations. The last operation STR (store) causes a return from the math routine with the program counter pointing at the next byte, which in this case is the RTS from the weight calculation. PI and RHO have to be defined in binary (hex) representation, which may require a separate program for calculating those representations when you write the assembler program. Note that I have not bothered to do that in the example program but just used dummy constants with all zero bytes.

Lastly, I've included a PL/9 program to do the same calculation. The saving in the actual calculation, CALCWT is obvious. PL/9 has the capability of reading the algebraic equation directly and generating the code necessary to perform the calculation (as would any Pascal or "C" compiler with floating point capability).

How do the amounts of code generated compare? It is very hard to say, because it is necessary to do a great deal of estimating in making the comparison. The BCD package byte count includes the routines to input and output numbers as ASCII strings. It comes to about 1500 bytes. The CALCWT portion in itself generated 92 bytes of code, the remainder being the math package.

The binary package with the routines necessary to

input and output numbers as ASCII strings would be about 1750 bytes, and the CALWT generated only 26 bytes of code. The 1750 is estimated. Actual code was about 1250 but that includes only being able to input and output numbers represented as +3141593-06 in ASCII form. That is, the handlers to convert to a number with embedded decimal point would have to be added, and of course the inverse function for input numbers, so I think the estimate is rather close.

The PL/9 version is unquestionably more straightforward than either of the others. It, however generated 2K of code with ASCII output capability, and 3K with ASCII input and output capability. The PL/9 package is more capable with regard to number I/O than either of the others. It will input ASCII with embedded decimal point, and will output in that form or revert to scientific notation if the number is too large or small. The calculation itself generated 44 bytes, considerably better than the BCD assembler version but not nearly as good as the package with the math interpreter. All the above byte counts include the complete four function math package with Add, Subtract, Multiply and Divide functions, since it would be rather difficult to subdivide the math package into four separate chunks that could be included optionally if they were needed, and most likely rather unnecessary, since any program doing any amount of calculation would use all four functions anyway.

Again, I must caution against the conclusion that the compiler version generates twice as much code as the assembler version. Let's assume that we wanted to do 100 calculations of the complexity of the one given here. The BCD package would then generate about 10.7K of code. The PL/9 version about 7.4K, and the binary about 4.7K. Compilers tend to look inefficient for small programs. Frequently they generate less "incremental" code than would be generated doing the job in assembler (compare 92 bytes for the calculation in BCD math package with 44 in PL/9).

Again I say that it is almost impossible to compare oranges with oranges in making these comparisons. These packages don't have the same capabilities, though I eliminated the scientific function code from all three for comparison. Perhaps another way to get a handle on the code generated would be to remove the ASCII input output capability of all three and compare. All three seem to come in around 1K of code for the test with those routines not included, that is with just the calculation and the floating point math capability.

I suppose there are some of you out there that will still argue that the assembler versions are clearer and easier to understand, or that the effort of writing such code is not that much more than the PL/9 version. You are going to have a very hard time convincing me of that at this point. I think I've shown here that the code efficiency can be better or worse one way or the other depending on the size of the program AND the particular math package or compiler that you use.

A Related Subject

I'd like to digress here for a minute and make a case for the use of Macros in assembler programming. You must have noticed the repetitive nature of the code for the BCD math package. Why write the same group of lines of code over and over again with different operands when you can do it with a few Macros? Of course you will have a macro definition library file that will be included with any calculation program, and the amount of code generated won't change, but the source listing becomes shorter. Listing WTBCDM shows two macro definitions, one using some conditional assembly, so that we can use one macro for multiplying two arguments from somewhere in memory, or can multiply the result of a previous calculation by a new argument. The macro is set up so that if the first argument is "*", the result of the previous calculation will be moved to the X register, and the second named argument will go to the Y register. Lastly the FPMUL will be done. The macro could be made more general at the expense of a little more typing, by including the operation in the arguments. We could write a macro CALC that would do for add, subtract, multiply, or divide, simply by making the first parameter, the operation. Rather than writing FMUL *,RADIUS,RESULT we would be able to do it by CALC FMUL,*,RADIUS or to add two numbers, CALC FPADD,PI,THREE,RESULT.

The listing shows how, with the macros that were defined above the calculation, CALCWT can be written in 6 lines. In order to see how well the macros have generated the required code, I've turned the macro expansion option on and assembled the program again as WTBCDMC. This listing shows the macro calls, and then lists the generated code. You will see that the code and byte count for all three versions of WTBCD is the same.

See, guys, I do know a little about assembler code, but I still would rather do it the easy way with a good compiler. By the way, I realize that I've not written the above code in position independent form. The original packages were not position independent, and I just fell in line with that approach in the examples, which are of course just examples anyway.

Keep the comments coming. This series is generating more response than anything that has ever appeared in this column. I guess we'll have to put all the columns with this material together in a book when we're done. Dan, you will have to have a share in the royalties on that one.

Reader Response

Bill Dickinson of Windrush wrote me a letter recently and he had a comment that covers a topic perhaps a bit broader than the present subject, but includes it. I will quote Bill directly.

"We are constantly dealing with people who are absolutely amazed at what we can do with the MC6809 and code produced by PL/9. There seems to be a lot of 'bigger is better' snobbery in the micro community that falls into the following groups:

1. If you are not using the latest 16/32 bit micro in your product, you can't be designing good products.

2. If your program is not written in assembly language it can't be fast.

3. If you use a high level language you NEED lots of RAM and ROM in the system. Therefore only the latest generation of 16/32 bit micros are suitable.

4. You NEED a processor with 1 MB + linear addressing to handle large programs and therefore the 8 bit micros can't do the job.

"My own limited experience with the micro community is that there are a lot of preconceived notions about the power of 8 bit micros out there! The worst of it is that the people who have formed these opinions are so convinced of the validity of their position that you can talk yourself blue in the face trying to argue the technical merits (or lack of them) of their position.

"We do not own or desire to own an emulator. Have you ever tried to convince someone that has used an emulator that it is possible to develop software and hardware without one?"

Note from Ron A.

We had an in circuit emulator with our first Exorciser system from Motorola. To my knowledge it was never ever used by anyone at the company, and I know for certain that I never used it. I have never had an "ICE" nor have I had a logic analyzer. I think the use of either would simply slow me down in the realm of my applications programs. I can see a need for a logic analyzer by people who write software in other realms. An in circuit emulator simply allows one to single step through a program in the "target system". The program should have been debugged before it got into the target system in the first place, by testing each routine with dummy data, and if necessary, single stepping with a tracer/debugger such as DEBUG from TSC.

Back to Bill's Letter

"From what I have seen of the code efficiency of the compilers available for the 68000 (et al) you NEED multi-megabytes just to cope with their gross inefficiency! The commercial boys and the large process control boys might need the amount of memory available with a 68000 but 99% of the control jobs we tackle can be done in well under 48K in PL/9 alone, and we do some very large machine tool and instrumentation control programs! My feeling about the 68000 in our line of work is that it is like using a 20mm machine gun to hunt deer ... it'll work BUT it is just too much machine for the job and has a lot of undesirable side effects."

Thanks Bill for those comments. My experience parallels Bill's directly. In one case, someone wrote me, sending an example of a single calculation using scientific functions via a BCD floating point math package (all programmed in assembler) with the comment that "This one equation runs for 25 seconds." My PL/9 coding of it took 33 milliseconds, about 750 times faster.

The point is that all assembly code is not guaranteed to be good, structured, fast, or efficient, and all compiled code is not guaranteed to be bad, slow, or inefficient.

That ends the quote from Bill Dickinson. I'd like to add that the comments regarding the efficiency of the compilers available for the 6800/09, sound like the ones I made a few years ago about the compilers available for the 6800/09. I firmly believe that it is possible to have very efficient compilers for the 68000, and that it is only a matter of time before they become available, just as the efficient Pascal, "C" and PL/9 compilers became available for the 6809. Bill did go on to add that he does believe that more efficient compilers for the 68000 are possible, and in fact that he is waiting for their appearance.

I was recently told by someone that he would use "C" for his 6809 development projects so that they would eventually be transportable to a 68000 system. I disagree. I won't ever want to move software that I have done and that runs adequately on a 6809, to the 68000. If I use a 68000, it will be for new applications that require its "power" and speed.

68000 Educational Module

The company bought one of these for me to look at. It arrived just today, and I don't have much to report this time. I have a cross compiler that will run on the 6809 system under FLEX, and I will be learning Assembler for the 68000 in the coming weeks and months. I firmly believe that a programmer ought to learn assembler for whatever processor he programs. It is always handy to know, and it helps a great deal in understanding the output of a compiler that generates assembler source code as an intermediate step in the compile process.

Don't be surprised if you see the prime number sieve benchmark here coded in 68000 assembler in a few months. Let's see if the 68000 is getting the bad reputation reported above in Bill's letter due to the instruction set, or due to inefficient software being available for it. Many of you readers may be too new to computing to remember the early reputation of the 6800 as a real "slug" because of the slowness of execution of the first BASIC that was available for it. (The BASIC interpreter was great, just slow).

```
/* CALCULATE WEIGHT OF A STEEL CYLINDER */

ORIGIN = $0;
STACK = $2FFF;


GLOBAL
  REAL
    HEIGHT,
    RADIUS,
    WEIGHT;

REAL PI 3.141593;
REAL RHO 4.528;


INCLUDE REALSHRT.LIB.1;
INCLUDE IOSHORT.LIB.1;


PROCEDURE CALCWT;
  WEIGHT = PI * RADIUS* RADIUS * RHO * HEIGHT;
ENDPROC; /* MAKES THIS A SUBROUTINE */
```

```
                    *
                    * CALCULATE THE WEIGHT OF A STEEL CYLINDER USING
                    * TSC BCD MATH PACKAGE ASSUMING 6809 VERSION AVAILABLE
                    *
                    * MATH PKG EQUATES
           2000 FPMUL   EQU   $2000
           2015 MOVE6   EQU   $2015
           3000 RSIGN   EQU   $3000
           3006 ISIGN   EQU   $3006
           300C YSIGN   EQU   $300C
                    *
                    * CONSTANTS
                    *
1000                      ORG   $1000
                    *
1000 00 03 14 15    PI    FCB   $00,$03,$14,$15,$92,$01 SIGN, 7 DIGITS, EXP
1004 92 01
1006 00 04 52 80    RHO   FCB   $00,$04,$52,$80,$00,$01 DENSITY OF STEEL
```

Flex User Notes listings

continues on page 47

OS9 USER NOTES

By: Peter Dibble
517 Goler House
Rochester, NY 14620

More About the CoCo Disk Driver

After sending in last month's column I had second thoughts about what I said about the OS-9 disk driver for the CoCo. I didn't believe what I had written. The gist of what I said was that Microware and Microsoft together were to blame for the non-standard disk driver included with the CoCo OS-9. The boot ROM in the CoCo loads just 15 sectors from track 34 on the boot disk into set locations in memory and jumps to them. This is Microsoft's idea of a nice way to boot a computer. What I said last month was that Microware managed to squeeze all of OS-9 into those 15 sectors by extensive compression of the code. This sounded pretty extreme to me, but I thought that was what I had heard from Ken Kaplan out at Microware.

Later, I became certain that I misunderstood Ken. There is no way all the core resident parts of OS-9 could be squeezed into that amount of disk, and, if all of OS-9 was loaded by the ROM boot, why does the CoCo have a two stage boot?

I called Microware to check my facts. I was wrong. In the first stage of the boot the CoCo ROM does load data from 15 sectors on track 34 into memory and jump to it, but only a few important parts of OS-9 are loaded: the kernal, the Init module, and the OS-9 bootstrap. These are the modules that are found in ROM on other OS-9 systems. The next stage of the boot uses the OS-9 bootstrap which was loaded in the first pass to do a normal OS-9 boot. The parts of OS-9 loaded in the first phase of the boot had to be squeezed hard, but much of the disk driver is loaded in the second phase of the boot.

There were a number of ways for Microware to get a full-featured disk driver into the CoCo, but they didn't. The restrictions on the first phase of the boot forced them to deviate from OS-9 standards in the boot module part of the disk driver. I believe they couldn't find a way to interest Tandy in the extra work (and memory) required to discard the boot after its work was done and load a driver that worked independently. That is certainly reasonable. Why should Tandy be interested in making it easy for people to use non-Tandy peripherals?

In any case, the problem seems to be solved. D. P. Johnson is advertising software that lets CoCo OS-9 deal with every disk format my Gimix can handle. I haven't tried his software, but I have heard from satisfied customers. I also own a 256K memory board made by Dan Johnson. I purchased one of the first boards he sold and had the kind of difficulties one might expect. I came to respect Dan Johnson while we struggled together to fix the problems which I discovered. He is good with hardware and software and VERY conscientious. I can't recommend the software because I haven't tried it (yet). I do recommend the man who sells it.

Where Next?

I have two very different OS-9 systems, a very large Gimix Level Two system and a CoCo. They fall at almost opposite extremes of the spectrum of microcomputers. The CoCo is so light and small that I think *nothing* of tucking it under my arm and walking a mile down to campus. The Gimix is so heavy that I am daunted by the thought of moving that stack of hardware even a few feet. The CoCo can't really handle more than one concurrent user. I routinely have two users on my Gimix and know people whose Gimix machines typically serve four or more concurrent users. The CoCo includes full graphics and a "terminal" protocol which is consistent across all CoCos. This is a big issue for other OS-9 users, particularly software developers who have to write programs which can be configured for any terminal.

Noting the similarities and differences between these computers has given me a lot of ideas about the kind of hardware I would like to see OS-9 running on. I imagine all computer users spend some time dreaming about the system they would have if only...

My dream computer is a personal computer, or, to use the popular phrase, a personal work station. I have grown used to the idea of OS-9 Level Two as a multi-user operating system, but I still prefer to think of it as a very powerful single-user system. Sharing computers is a way to save money. When I imagine the computer I would like, I don't consider money first.

Naturally, my dream computer runs OS-9 Level Two. It includes a bit-mapped screen (color optional), several dedicated processors, support for some graphics input device (I haven't chosen between a bit pad, a mouse and a light pen), and more than plenty of memory.

Many people seem to think that 128K is the right amount to run OS-9 Level Two in. Now you CAN run Level Two in even less, but you don't really appreciate it until you get to at least 192K. My dream machine would have at least 192K upgradable to 256K, better still, 512K. There are so many uses for memory! Solid state disk drives or caches give better access times than hard disks but use a lot of memory. Complex programs can take lots of memory, but, when they are well written, they are powerful and easy to use. Sometimes lots of memory is needed for simple storage of data. I know a woman who keeps running out of space for her spread sheet on an IBM PC. She has about 600K! So lets put lots of memory in the dream machine.

Graphics hardware is never good enough. At any rate that's the way I react to it. If the resolution and the number of colors is sufficient, the screen takes too long to update. If data is displayed by fussing with parameter lists and registers, the system is too limited. If the screen is bit-mapped, it takes too much attention from the CPU to control the screen. The best solution seems to be to have a separate processor that deals with a bit-mapped display. If the graphics processor has a very high speed connection to the rest of the system, and can be dynamically programmed to do more than just update the screen, the result should be speed and flexibility in graphics.

There is use for more than one special processor in my dream computer. If graphics support is included in the package, it would be foolish to require a terminal to be attached to the computer; an attached keyboard would be sufficient. A dedicated pr essor to scan the keyboard would take another load off the main processor. The other I/O devices could also use their own processors. My Gimix uses a 6809 on one of its serial cards to take some of the interrupt load off the main processor. It speeds my machine up a little, but doesn't have any other use. If the software for the I/O processor was loaded (and reloaded) by the main processor it would let the serial board be programmed to handle high-speed networks and other applications where timing is important. Even disk controllers could use their own special processors. I don't know of any programmable disk controllers that could do for disk I/O what smart serial cards has done for terminal I/O. The Gimix Intelligent serial card contains a good part of SCFMAN. By unloading this work onto a special processor more cycles are left for user programs. RBFMAN is more complicated than SCFMAN and uses more CPU time. If most of that work could be done by a separate processor still more of the resources of the main processor would be available for the user.

In fact, why talk about the main processor? In many cases OS-9 processes don't share memory with one another. If the dream computer had a bus where additional processor boards with some memory and perhaps I/O could be inserted, OS-9 could run independent processes on their own procedures. Most personal work station users don't need to run more than three or four processes at a time, so including many of what amounts to separate computers in the package would be wasteful. But, if the power is available the applications will arrive.

Mice are making a big splash these days. The Xerox Star, the Apple Macintosh and Lisa, and lots of more expensive work stations are using them. I would definitely pick a mouse over a joy stick. I have more trouble deciding that a light pen or graphics pad isn't a better tool than a mouse. The graphics pad is very precise and the stylus can be used about like a mouse. The arguments against graphics pads are that they are expensive, require desk space, and, for some applications, force the person using them to mentally map from the bit pad to the screen. The cost problem I will ignore -- after all this is a dream computer. The other two problems apply to mice as well. A light pen doesn't require desk space or a mental mapping, but I don't find

them very precise and my hand obscures the screen when I am pointing. I can't make up my mind.

A fancy computer like this deserves fancy software. The peanut-butter and jelly programs now available for OS-9 just don't live up to the hardware.

My pet peeve with OS-9 software has always been its lack of excellent editors. I like Dynastar fine, and I have heard nice things about Scredítor and Stylograph, but these programs are at least five years behind the state-of-the-art. My dream machine deserves something special. Do you suppose EMACS could be ported to OS-9?

A real database program would be nice. Something more than a filing cabinet or stack of index cards metaphor.

I bet Knuth's TeX would run on something like this. Some good graphics programs, especially a graphics editor would make the graphics support a lot more useful. A real statistical program like SAS, or SPSS would make some people happy. Others need really good communications software.

Languages aren't as important as the software written in them, but OS-9 is still painfully short of languages. I bet APL would run well under OS-9. Fortran is old fashion, but we really should have it. Those are the fundamental languages, but there is an endless list, including: Pilot, PL/1, Logo, Smalltalk, and others.

Networking is another sexy topic these days. Expensive computers (which my dream machine is turning out to be) are generally used by people for whom communication is terribly important. Electronic mail, electronic calendars, and sharing of files and other resources are important to them. OS-9 doesn't include networking software, but I think it will be at least as easy to run over a network as any other operating system.

Enough of the dreaming. Truly, my dream machine is not so very far away. I/O processors exist, and I am sure more are coming. I have heard talk about slave processors. There are graphics boards available for the SS-50 bus that are a lot like what I have in my dream machine. The CoCo comes with bit mapped graphics standard.

For my Gimix I can hope for I/O and slave processors and a better (and less expensive) graphics board. For my CoCo I can aim low and hope for a disk controller with an onboard buffer, or aim high and look for a real Level Two system with as much done in hardware as possible (I/O, sound, and graphics). From my viewpoint as a Level Two user I think Tandy would be crazy not to offer a CoCo with Level Two. For a software person like me, it is fun to think up lots of things that hardware people should do for us, but the most important part of any computer is its software.

Some of my software wish list will have to wait for better hardware, in particular for more memory. Much of it can be done now. I have done some primitive networking myself. A really special database program or editor would push a 6809 hard, but might be possible. I have heard from people who are working on lots of nice things for OS-9. Pretty near every piece of software for my dream machine is a project someone is working on now.

More Noise from the CoCo

Last month I included a driver for the Digital-to-Analog converter in the CoCo. That driver was useful for low-speed D/A applications, but it didn't do very well at sound generation. The highest pitch my driver could manage was something of a gurgle. The speed problem wasn't in the driver. It takes a long time for a character to get through SCFMAN. Even when a block of characters goes through together there is enough delay in the transmission of each character to make smooth, high-frequency waves impossible. Fortunately, generating music isn't the only purpose for an A/D converter. Controlling lab instruments, motors, and such are all fine applications which only require a voltage to be changed infrequently -- 10 times per second at most.

I ended last month's column with a few suggestions for ways to make the A/D driver better at generating sound. This month I went ahead and took my suggestions. This month's A/D driver does a pretty good job of making music. It even makes nice chords. I made the improvement I suggested last month. If the driver receives a zero, it places the next 360 bytes sent to it in a special buffer. Characters that don't go into the buffer cause the contents of the buffer to be transmitted through the D/A a number of times corresponding to the magnitude of the character

written. Since it takes a fixed amount of time to transmit the buffer, each character from $01 to $FF will take a fixed amount of time to send. This way each character sends a note of a set duration whatever the pitch.

At first I used a buffer 128 bytes long. That was easy to handle in BEEPER, but it was hard to build a wave in. It is important that a whole number of cycles fit into the buffer. It was difficult to generate a wave that fit precisely into 128 values. Numbers like 90 and 360 work better when angles are measured in degrees (if they are measured in radians it is hard to make any integers come out evenly.) I tried a 90 byte buffer, but I found it hard to store smooth, high-pitched tones in it. After the buffer got over 128 bytes long, I used the D register to offset the index into it so length didn't make much difference. I chose 360 bytes as the length of the buffer because it is an easy number to work with when generating the wave.

Interrupts are a problem to time-dependent things like music. I tried BEEPER with interrupts masked and unmasked. When interrupts are unmasked the sound is definitely not pure; however, when the interrupts are masked lots of bad things happen. With interrupts masked nothing happens except operation of the D/A. Time doesn't get updated, the keyboard doesn't get scanned, and, if you are using the RS-232 port, it comes to a halt. In the version of BEEPER included with this column I commented out the ORCC and ANDCC. Try it both ways and choose for yourself.

This Month's Driver

I got a little carried away with the test driver for BEEPER. The program calls for a magnitude and frequency for a wave (the numbers are only relative). The sine wave generated with these numbers is added to whatever wave has already been generated. The resulting wave is displayed. If a Y is entered, the wave is loaded into BEEPER's buffer and a few beeps are sent, if an A is entered another sine wave is prompted for and added to the existing wave, and if anything else is entered the wave is erased and the program starts over with a clean slate.

I am afraid that this test driver is another program that needs work. BEEPER truncates numbers greater than 63 to 63. If the sum of the sine waves loaded into BEEPER's buffer is greater than 63 at any point the wave will be clipped (as hi-fi people say). It would be good if TBEEP2 would check for this. I also get pretty frustrated when I don't like the last sine wave I added to a wave I am building and have to wipe out the entire waveform to get rid of it. On the other hand I am rather partial to the graphic display of the waveform.

The Users Group

Things aren't moving as quickly for the OS-9 Users Group as we hoped they would. We published our first news letter (called MOTD) months ago. By the way, if you are a member and didn't receive a copy of MOTD, send a note to the Users Group. Our system for keeping track of members seems pretty reliable, but it may have cracks in it. We are working on the second issue. The most important thing to most members seems to be the software exchange. There have been a number of problems getting the software exchange disks out. The most interesting problem has been a disk incompatibility between 40 track disks written by 80 track drives on two different manufacturer's systems. Watch for this problem!

There also seems to be some trouble getting disks. Three dollars per disk delivered to a member is a very low price. It's hard to be too impatient. In any case, barring another serious hold-up, the disks should be in the mail by late March. Let me say again that we don't know the disk format many users need. If I guess wrong, send the Users Group a letter and we'll try to find a way to straighten things out.

```
00001                          NAM   BEEPER
00002                          IFP1
00003                          ENDC                Use OS9DEFS, SCFDEFS and IODEF
00004                                              and ENDC
00005                          USE   BEEP
00006                          TTL   DEVICE DESCRIPTOR
00007                          NAM   BEEP
00008  00F1            TYPE    SET   DEVIC+OBJCT
00009  0000 87CD0027           MOD   BPEND,BPNAM,TYPE,REENT+1,FMANAME,DRVNAM
00010  0009 03                 FCB   READ.+WRITE. MODES
00011  000E FFFF20             FCB   $FF,$FF,$20 PORT ADDRESS
00012
00013  0011 01                 FCB   OPTL          Length of options section
00014  0012                    OPTIONS EQU   *
00015  0012 00                 FCB   DT.SCF
00016  0001                    OPTL  EQU   *-OPTIONS
00017
00018  0013 FF23               CNTL1 FDB   $FF23     address of control byte 1
00019  0015 FF03               CNTL2 FDB   $FF03     address of control byte 2
00020  0017 42454550           BPNAM FCS   /BEEP/    name of this module
00021  001B 534A3C6            FMANAME FCS   /SCF/     File Manager name
00022  001E 42454550           DRVNAM FCS   /BEEPER/  Device driver name
00023  0024 58AEA3                   EMOD
00024  0027                    BPEND EQU   *
00025                          TTL   DEVICE DRIVER FOR D/A
00026  00E1            TYPE    SET   DRIVR+OBJCT
00027  0081            REVS    SET   REENT+1
00028  0000 87CD0027           MOD   BPEND,BPNAM,TYPE,REVS,ENTER,MEMSIZE
00029 D 001D                    ORG   V.SCF
00030 D 001D            PORTA   RMB   2             PORT ADDRESS
00031 D 001F            CTL1V   RMB   1             HOLD CNTL1 VAL
00032 D 0020            CTL2V   RMB   1             HOLD CNTL2 VALUE
00033 D 0021            CTL1A   RMB   2             HOLD CNTL1 ADDR
00034 D 0023            CTL2A   RMB   2             HOLD CNTL2 ADDR
00035 D 0025            COFFSET RMB   2             OFFSET IN BUFFER
00036  0168             BUFLEN  EQU   9044
00037 D 0027            BUFFER  RMB   BUFLEN
00038 D 018F            MEMSIZE EQU   .
00039  0000 03                 FCB   READ.+WRITE. DRIVER MODE
00040  000E 42454550           FCS   /BEEPER/      Program Name
00041  0014 01                 FCB   1             EDITION
00042                          **********
00043                          * Entry points
00044                          *
00045  0015                    ENTER
00046 U 0015 160006             LBRA  INIT
00047 U 0018 160042             LBRA  READ
00048 U 001B 160047             LBRA  WRITE
00049  001E 160080             LBRA  GETSTAT
00050  0021 16009B             LBRA  PUTSTAT
00051  0024 1600FC             LBRA  TERM
00052  0027                    INIT
00053                          *******
00054                          * U ADDRESS OF DEVICE STATIC STORAGE
00055                          * Y ADDRESS OF DEVICE DESCRIPTOR MODULE
00056                          *
00057  0027 AEA813             LDX   CNTL1,Y       Move the address of cntl1 byte
00058  002A AFC821             STX   CTL1A,U       from the D.Descriptor to stati
00059  002D A684               LDA   ,X            save the value of the cntl1 by
00060  002F A7C81F             STA   CTL1V,U
00061  0032 8A08               ORA   #$08          set one of the bits
00062  0034 A784               STA   ,X            that turns on sound
00063  0036 AEA815             LDX   CNTL2,Y       Move the address of cntl2 byte
00064  0039 AFC823             STX   CTL2A,U       from the D.Desc to static stor
00065  003C A684               LDA   ,X            save the value of the cntl2 by
00066  003E A7C820             STA   CTL2V,U
00067  0041 84F7               ANDA  #$FF-$08      set the other bit
00068  0043 A784               STA   ,X            that turns on sound
00069  0045 8D08               BSR   INITBUF       Initialize the sound buffer
00070  0047 1C                 CLRB                CLEAR CARRY
00071  0048 E7C825             STB   COFFSET,U     Coffset is a two byte field
00072  004B E7C826             STB   COFFSET+1,U
00073  004E 39                 RTS                 RETURN
00074                          ************
00075                          *  Put something that won't sound too bad
00076                          *  into the sound buffer
00077                          *
00078  004F                    INITBUF
00079  004F 30C827             LEAX  BUFFER,U
00080  0052 CC0167             LDD   #BUFLEN-1
00081  0055                    INITLOOP
00082  0055 E788               STB   0,X
00083  0057 830001             SUBD  #1
00084  005A 2CF9               BGE   INITLOOP
00085  005C 39                 RTS
00086  005D                    READ
00087                          *********
00088                          * U ADDRESS OF DEVICE STATIC STORAGE
00089                          * Y ADDRESS OF PATH DESCRIPTOR
00090                          * RETURN CHARACTER READ IN A
00091                          *
00092  005D AE41               LDX   V.PORT,U
00093  005F A684               LDA   ,X            set the value in the D/A regis
00094  0061 44                 LSRA
00095  0062 44                 LSRA                Shift out the low order bytes
00096  0063 1C                 CLRB                Clear carry
00097  0064 39                 RTS
00098  0065                    WRITE
00099                          ********
00100                          * U DEVICE STATIC STORAGE
00101                          * Y PATH DESCRIPTOR
00102                          * A VALUE TO WRITE
00103                          *
00104  0065 6DC826             TST   COFFSET+1,U If coffset isn't zero
00105  0068 263F               BNE   DEFINE        we are in the process of fillg
00106  006A 6DC825             TST   COFFSET,U     buffer.  We have to tst both C
00107  006D 263A               BNE   DEFINE
00108  006F 4D                 TSTA                If the character to write is 0
00109  0070 272F               BEQ   SOUNDIT       the sound buffer
00110                          ********
00111                          * LOOP THROUGH BUFFER
00112                          *
00113  0072 30C827             LEAX  BUFFER,U      The address of the sound buffe
00114  0075 3402               PSHS  A             SAVE COUNT
00115                          * ORCC #INTMASK Shut off interrupts
00116  0077                    CYCLE
00117  0077 CC0167             LDD   #BUFLEN-1     Offset in buffer
00118  007A 3406               PSHS  D             Save offset
00119  007C                    WLOOP
00120  007C A6DB               LDA   D,X           set a byte out of buffer
00121  007E 3410               PSHS  X
00122                          **** on second thought it would have been
00123                          **** better to just do a leax BUFFER,U later instead
00124                          **** of saving this value here
00125  0080 AE41               LDX   V.PORT,U      The address of the D/A registe
00126  0082 3402               PSHS  A             Build the byte to store in the
00127  0084 A684               LDA   ,X            register
00128  0086 8A00               ORA   #%00000000
00129  0088 A6E0               ORA   ,S+
00130  008A A784               STA   ,X            Store the new D/A value
00131  008C 3510               PULS  X             recover the buffer address
00132                          ***** see note
00133  008E E0E4               SUBB  ,S            set the new offset in buffer
00134  0090 830001             SUBD  #1            decrement the offset
00135  0093 E0E4               STD   ,S
00136  0095 2CE5               BGE   WLOOP         if it isn't negative send the
```

```
00137  0097 3262              LEAS  2,S       Clear stack
00138  0099 6AE4              DEC   ,S        decriment repeat count
00139  009B 26DA              BNE   CYCLE     cycle if not zero
00140       * ANDCC #$FF-INTMASKS
00141  009D 3261              LEAS  1,S       CLEAR STACK
00142  009F 5F                CLRB
00143  00A0 39                RTS
00144
00145  00A1            SDEFINE
00146  00A1 C001A8            LDD   #WAVELEN
00147  00A4 EDDC25            STD   COFFSET,U
00148  00A7 1C                ANDCC #$FE      CLEAR CARRY
00149  00A8 39                RTS
00150       ************
00151       * Load the Sound buffer
00152       *
00153  00A9            DEFINE
00154  00A9 48                LSLA            Prepare the value
00155  00AA 48                LSLA
00156  00AB 3402              PSHS  A         save it
00157  00AD EADC25            LDD   COFFSET,U Current offset
00158  00B0 830001            SUBD  #1
00159  00B3 EDDC25            STD   COFFSET,U Update offset
00160  00B6 30CBC7            LEAX  BUFFER,U
00161  00B9 308B              LEAX  D,X       location to store this byte at
00162  00BB 3502              PULS  A         get the byte
00163  00BD A784              STA   ,X        store it
00164  00BF 5F                CLRB
00165  00C0 39                RTS
00166  00C1            GETSTAT
00167  00C1            PUTSTAT
00168  00C1 5F                CLRB
00169  00C2 39                RTS
00170  00C3            TERM
00171       ********
00172       * U DEVICE STATIC STORAGE
00173       *
00174  00C3 AEC821            LDX   CIL1A,U
00175  00C6 A6C81F            LDA   CTL1V,U
00176  00C9 A784              STA   ,X        restore the original ctl1 valu
00177  00CB AEC823            LDX   CIL2A,U
00178  00CE A6C820            LDA   CTL2V,U
00179  00D1 A784              STA   ,X        restore the original ctl2 valu
00180  00D3 5F                CLRB
00181  00D4 39                RTS
00182  00D5 A6ADDEAD          EMOD
00183  00D8            EPGMEND EQU   *

00000 error(s)
00003 warning(s)
$00FF 00255 program bytes generated
$0172 00370 data bytes allocated
$1602 05842 bytes used for symbols
```

```
PROCEDURE TBEEP2
0000      (* ---------------------------------------------------
0032      (* TBeep2 is a test driver for the device driver
0062      (* BEEPER.  It loads BEEPER with a wave form,
008F      (* then sends it a few more characters to test the tone.
00C7      (* ---------------------------------------------------
00F9      (*
00FC      (* Note is an array which contains the values which will be sent to
013F      (* the D/A to form a note
0158      (*
0158      DIM NOTE(360):BYTE
0167      DIM I,J,K:INTEGER
0176      DIM SOUND:INTEGER \(* Path number for A/D
0193      DIM MAGNITUDE,FREQ:INTEGER \(* variables used to form the waveform
01C4      DIM C:BYTE \(* a utility one-byte variable
01E9      DIM CMD:STRING \(* waveform command
0203
0204      OPEN #SOUND,"/BEEP":WRITE
0214      DEG  \(* Use degrees for angles
0225      (* Initialize Note to zeros
0244      FOR I=1 TO 360
025B        NOTE(I)=0
0266      NEXT I
0271
0272      (*
0275      (* Build waveform
0286      (*
0289      LOOP
0288        RUN GFX("ALPHA") \(* make screen printable
02B0        (* Get parameters for a sin wave
02DD        INPUT "MAGNITUDE* *.MAGNITUDE
02E3        INPUT "FREQUENCY: *.FREQ
02F6        (*
02F9        (* add the sin wave to the wave in NOTE
0320        (*
0323        FOR I=1 TO 360
033A          NOTE(I)=NOTE(I)+MAGNITUDE*SIN(I*FREQ)
0358        NEXT I
0363        (*
0366        (* Display the graph
0378        (*
0371        RUN GFX("MODE",0,1)
038F        FOR I=1 TO 180
039F          J=NOTE(I*2)-2
03B0          K=J+4
03B8          IF J<0 THEN J=0
03CD          END IF
03DF          J=J*2
03DA          K=K*2
03E5          IF J>192 THEN J=192
03F7          END IF
03F9          IF K>192 THEN K=192
0408          END IF
0408          RUN GFX("LINE",I,J,I,K)
0420        NEXT I
0438        (*
0438        (* Display a little bit of the next cycle
0464        (* to demonstate the the wave is continuous
0485        (*
0492        FOR I=181 TO 255
04A2          J=NOTE((I-180)*2)-2
04B6          K=J+4
04C1          IF J<0 THEN J=0
04D0          END IF
04D5          J=J*2
04E0          K=K*2
04EB          IF J>192 THEN J=192
04FD          END IF
04FF          IF K>192 THEN K=192
0511          END IF
0513          RUN GFX("LINE",I,J,I,K)
0533        NEXT I
0538        (*
0541        (* There is no prompt because the screen is full of
0576        (* graphics, but enter YCORD #CORD, or #CORD after
05A6        (* the graph has been drawn
05C1        (*
05CA        INPUT CMD
05D9      ELSIF CMD="Y" THEN
05DA      END IF
05DA        IF CMD<>"A" THEN
05E7          FOR I=1 TO 360 \(* The waveform is bad.
0610            NOTE(I)=0 \(* zero it and start over
0634          NEXT I
063F        END IF
0641        RUN GFX("CLEAR")
064E      END LOOP
0652      RUN GFX("ALPHA")
065F      RUN GFX("QUIT")
0668      C=0 \(* a zero tells the driver to use the next 360 characters
0688      PUT #SOUND,C \(* to build a new form
068D      PUT #SOUND,NOTE \(* send the new form
06E9      PRINT "STARTING SOUND"
06FB      (*
06FE      (* Send a few beeps of different lengths
0726      (*
0729      FOR I=100 TO 250 STEP 50
073E        C=I
0744        PUT #SOUND,C
0750        #OSUB 100
0754        PRINT "END OF LOOP ";I
0768      NEXT I
0773      END
0775      (*
0778      (* Delay a little
0789      (*
078C  100
0790      FOR J=1 TO 500
07A1      NEXT J
07AC      RETURN
```

OS9 USERS GROUP

OS-9 Users Group
Presidents Column
by Dale Puckett

Before I start answering questions I would like to thank Jim Bellomo for making the live Compuserve conference with the OS-9 Users Group officers February 12 possible. Without his hard work we wouldn't have this fantastic communications medium available to us. ** This was written prior to the untimely death of Jim Bellomo, noted in last months 68 MICRO JOURNAL. **

And, while we're passing out KUDOs I also would like to thank Tom Murphy, of SunTel in St. Louis, for all the work he did as Secretary after our election in Des Moines, last August. Tom resigned so he could personally supervise a new and rather large contract. Good luck to you Tom!

I have appointed Dave Gibson, of the Federal Highway Administration in Washington, D. C. to fill the rest of Tom's term of office.

WHO ARE WE AND WHAT ARE WE DOING FOR YOU?

We are a Des Moines, IA based corporation formed to promote the use of computers in general and Microware's OS-9 operating system in particular. This year's officers were elected at the Second Annual Microware OS-9 seminar during August, 1983. They were given a mandate to incorporate the group, establish a method of communication and set up a software exchange. Progress is being made in all areas. However, to mimic an old saying, four officers does not a dynamic group make.

WHAT CAN YOU DO FOR US?

You, the OS-9 user, are the lifeblood of the group. If you take an active part in the

organization by contributing programs to the software exchange, articles for the newsletter and technical help for the newcomers, we will some day be a viable force in the industry.

WHY SHOULD WE CARE?

The 6809 microprocessor -- the best on the market -- has run behind all others in the personal computing field since the beginning because there has been no coordination and cooperation. With the increase in the number of users made possible by Tandy's Color Computer version of OS-9. we again have a chance to become a viable force in the market place.

WHAT KIND OF HELP DO WE NEED?

a. We need engineers and systems programmers to help George get the bulletin board system on line.

b. We need recruiters and personnel types to help Peter increase the membership.

c. We need people to help Dave Gibson put out the newsletter. We need people to manage the software library.

d. We need others to write articles and stir up publicity.

e. And, we need people to coordinate the exchange of information between the many local OS-9 users groups we hope to inspire.

Please, say you want to help. Send us a letter and tell us what you would like to do. We'll put you in touch with the proper committee chairman immediately. And remember if you solve a problem or create something you're proud of, send it to us for publication in MOTD and leave a note here on the Compuserve OS-9 SIG.

WHAT IS YOUR MEMBERSHIP POLICY?

Your membership will be good through January 1, 1985. Renewals at or after the Third Annual OS-9 Seminar in August 1984, will be good from then through December 31, 1985. All members, regardless of when they join, will receive all services of the Users Group provided since August, 1983. All OS-9 Users Group members will have access to the special Compuserve XA-4 data base which contains our complete software exchange library.

WHAT ARE YOUR DUES?

Every member pays $25.00 per year. However, since there was very little activity during the first year the group existed and since we got off to a slow start this year, anyone who has joined in the past will not have to pay dues again until 1985.

WHAT DO I GET FOR MY MONEY?

We are trying to push forward on several front. I feel that the most important thing we are doing for you is maintain an excellent software exchange library. Dave Kaleita has done a terrific job so for. Then, it will be up to you the members to keep contributing.

I also hope we will be able to publish a few more news letters this year. We have planned to make the next one a technical issue filled with questions and answers. You should see it by the end of April. Please. If you have a hot bit of news you think everyone should know. Send it to:

MOTD
OS-9 Users Group
P. O. Box 8027,
Des Moines, IA 50301

HOW DO I LET YOU KNOW I WANT TO HELP?

Send a letter to me at the address above. Please mark the outside of the envelope, "Volunteer." I will then contact the proper chairman and he will get in touch with you.

HOW ARE YOU GOING TO HANDLE
THE INFLUX OF COCO OS-9 USERS?

Bringing the CoCo OS-9 user on line and up to speed is one of our most important priorities. The more we can help them prosper, tha faster OS-9 will grow in popularity. That means it will hold a more dominant position in the marketplace and there will be more software available for all of us.

HOW ARE YOU ORGANIZED?

We have already set up several standing committees. Unfortunately, we have almost no one to fill the positions. However, I have defined two more that we will set up as soon as we have people to fill the positions.

Presently, the only committee chairman that is active is Dave Kaleita who heads our software exchange efforts. So far it has been a committee of one.

Kudos to "Van" and "Jack" who have volunteered to work with Dave to organize our CoCo software holdings. Thanks! This is the kind of spirit we need. If this goes the way we think it will ... Dave, Van and Jack are going to need a lot more help. Please send your "volunteer" letter.

WE NEED A Communications CHAIRMAN?

Our communications committee chairman has disappeared into the wordwork and George Dorner, our fantastic treasurer has had to struggle with the problems alone. Please, help him.

We have set a goal to get our OS-9 Bulletin Board on the air this year. The equipment, donated by Richard Don at GIMIX, is already on line. Now all we need are the engineers. If you live in the Chicago area -- anywhere for that matter -- and can help, please call George.

Our end goal is to have a bulletin board that looks and acts like OS-9.

WE NEED A MEMBERSHIP COMMITTEE?

So far, our vice president, Peter Dibble, has had to handle the membership committee chores. If you are good with people -- an Army recruiter maybe -- drop us a line. Please!

WHAT OTHER COMMITTEES WILL YOU BE ADDING?

a. I plan to appoint a committee to coordinate local club activity. This is an area where we can really help the CoCo crowd. This will increase the exchange of information and bring the help of the oldtimers to our new found friends.

b. I plan to appoint an education committee. This committee will help prepare the question and answer section for our newsletter. I also plan to have them work with Jim Bellomo and hope we will be able to put an OS-9 Q & A help data base here on the OS-9 SIG. This new committee will work out the details.

c. I need a Chicago area volunteer(s) to serve as an interface between the Compuserve OS-9 SIG and the OS-9 Bulletin Board. If you can handle it, drop me a line.

d. I plan to appoint a by-laws committee -- all officers will be a member also -- to study several issues. The most important is the election of officers.

WHAT'S THE BOTTOM LINE?

Remember, if you do nothing, the group will not live long, and you will get nothing from it. It's all up to you.

- - -

READING NON-FLEX BASED DISKS

Reading Non-Flex Based Disks

How many time have you wanted to use your Flex system to read and copy disks formatted on other non-compatible operating systems? If you are like me and have access to disks formatted under CP/M, NEWDOS-80 or SSB's DOS69D, then you will find the following three transfer utilities very useful indeed.

Even if you have no use for such transfer functions, some of the coding routines and techniques used by these utilities are invaluable for those of you who write your own Flex assembler programs.

I have included three programs; one which transfer TRS-80 NEWDOS-80 V.2 formatted data to Flex; and one which transfer CP/M formatted data to Flex. All three are quite similar in terms of the subroutines used to implement them. The difference between them lies of course in the code used to inte pret each of their disk formats.

The common routines used by all three utilities are READSS, MCV, CLC, PDATA, ROUTE, PRTSET, and RESET. Before explaining the usefulness of some of these subroutines to the Flex programmer, I would like to describe the operations of the transfer programs in general.

All three programs allow the user to view the particular disk's directory of files and transfer any existing file on that disk to FLEX (or to your terminal or printer). Each has a similar menu describing the possible functions. A help selection has been included for the CPM and DOS utilities. This selection describes the features of the program. Since the basic facilities implemented in all three programs are the same, the help selection for the TRS utility has been omitted.

Let us now look at the limitations of each utility in turn.

The CPM-to-FLEX utility was tested using disks from the "C User's Group" and from the Osborn system. It will handle both 5" and 8" disks to the extent outlined in the program code.

The DOS-to-FLEX utility was the easiest to implement because it required no special mapping functions to read the data. Since I also run DOS69D on my 6809 system, I was able to write the program with few problems.

The TRS80-to-FLEX utility was the most difficult to implement because the disk data structure used by NEWDOS-80 is quite different from what I am used to. The concept of granules, lumps, GATs, HITs and FDEs is anything but straightforward. The documentation included at the beginning of the program explains most of the assumptions made in the code. In addition, the utility will only handle files having no more than four space areas assigned to it (in other words, it has no extents).

Now let us take a look at some of the subroutines common to the utilities, specifically, ROUTE and PRTSET.

These two functions illustrate how to use FLEX resident routines to make the routines you write more FLEXible (ha ha ha). The ROUTE subroutine allows output to be routed to a terminal, printer or file, by using OSWTCH in conjunction with the OUTCH, OUTCH2 and PUTCHR routines supplied by FLEX. This is how the routine works (description taken from TSC's FLEX Programmers Manual, 1978).

If the output switch (OSWTCH) is zero, output performed by the PUTCHR routine is through OUTCH and if non-zero OUTCH2 is used. The OUTCH routine usually does the same as OUTCH2, however, OUTCH may be changed by programs to refer to some other output routine (such as one to drive a printer). OUTCH2 is never changed.

By using PUTCHR to output all "routeable" information, we now have a mechanism to change the route very easily. If the output switch is non-zero, the routine OUTCH2 is used by PUTCHR to send the character. If zero, PUTCHR checks the location "File Output Address (FOA)". If it is non-zero the contents of this location is used as an address of an FCB of a previously opened-for-write file, and the character is written to the file. If zero, the routine OUTCH is called to process the character. Normally, OUTCH sends the character to the terminal. The user may, however, change the address portion of the OUTCH entry point to go to another character output routine (such as a printer output routine).

By using ROUTE along with PRTSET and RESET, the user can enhance any FLEX utility to allow routeable output. PRTSET is used to load the printer output routine if it has not already been loaded. RESET is needed to reset the output switch and close any file that might have been opened for output. It is called at the end on each selection function (COPY and DIR).

The reader will also notice the use of DSECTS and dynamic data areas (using the system stack) in the utilities. Study how these structures are used to make coding easier and more readable.

For those readers who do not wish to type in all the code, I can supply you with Xerox laser printed listings of all the programs for $5.00 (to cover postage and handling). Or, if you want the source code to some or all of the utilities on disk, send $4.00 per program, $8.00 minimum.

Happy Programming !

Editor's Note: Due to the extensive size of the assembled source listing of all three major programs, they will be continued over a period of 3 or 4 months, as available space permits.

Readers interested in not waiting or desiring not having to type in the large amount of code should take advantage of the printout or disk offer above.

DMW

- - -

```
*********************************************************
*                                                       *
*    CP/M to FLEX Utility                               *
*                                                       *
*       Written by: Scott R. Fraser                     *
*                   547 Sharron Bay                     *
*                   Winnipeg, Manitoba, Canada          *
*                   R2G 0H8                             *
*                   Ph. (204) 338-7641                  *
*                                                       *
*********************************************************


            * Define CP/M directory dsect
            *
0000                ORG   $0000
            0000 CPMDIR EQU   *
0000        CPMFEF  RMB   1       file entry flag
0001        CPMNAM  RMB   8       1st part of name
            0008 CPMNL  EQU   *-CPMNAM  len of this part
```

```
0009                CPMEXT  RMB   3         extension
            0003    CPMEL   EQU   *-CPMEXT  length of extension
            0008    CPMNL   EQU   *-CPMNAM  len of entire name
000C                CPMCF   RMB   1         continuation flag
000D                CPMRS2  RMB   2         unused
000F                CPMSC   RMB   1         file sector count
0010                CPMBLK  RMB   16        list of blocks
            0010    CPMBS   EQU   *-CPMBLK  # blocks per file entry
            0020    CPMLEN  EQU   *         length of CPM dsect

0100                        ORG   $0100
                    *
                    ***** DEFINE PROMPTS USED BY ALL ROUTINES
                    *

                    *
                    * PDIR routine
                    *
            0100    HEAD    EQU   *
0100 0D0A                   FDB   CRLF
0102 4E 41 4D 45            FCC   /NAME    TYPE   SIZE  BLOCKS/
0110 20 20 20 45            FCC   /  EXTENTS/
0127 0D0A                   FDB   CRLF
0129 04                     FCB   EOT

            012A    WELCOM  EQU   *
012A 0D0A                   FDB   CRLF
012C 44 65 73 63            FCC   /Description of Selections:/
0146 0D0A 0D0A              FDB   CRLF,CRLF
0148 20 20 20 30            FCC   /   0 - you're looking at it buddy!/
016C 0D0A                   FDB   CRLF
016E 20 20 20 31            FCC   /   1 - this selection will yield a CPM /

0195 64 69 72 65            FCC   /directory listing./
01A7 0D0A                   FDB   CRLF
01A9 20 20 20 20            FCC   /       The user is prompted for the drive/
01D2 20 6F 66 20            FCC   / of the CPM disk/
01E2 0D0A                   FDB   CRLF
01E4 20 20 20 32            FCC   /   2 - this selection will copy a /
0206 67 69 76 65            FCC   /given CPM file to a/
0219 0D0A                   FDB   CRLF
021B 20 20 20 20            FCC   /       FLEX file.  The user is/
0239 20 70 72 6F            FCC   / prompted for the "from" drive/
0257 0D0A                   FDB   CRLF
0259 20 20 20 20            FCC   /       (ie. where the CPM disk is)/
0278 20 61 6E 64            FCC   / and the "from" file/
028F 0D0A                   FDB   CRLF
0291 20 20 20 20            FCC   /       name.  No conversion (say/
02B1 20 6C 6F 77            FCC   / lowercase -> uppercase/
02C9 0D0A                   FDB   CRLF
02CA 20 20 20 20            FCC   /       for example) is done on this/
02ED 20 6E 61 6D            FCC   / name, so type it/
02FE 0D0A                   FDB   CRLF
0300 20 20 20 20            FCC   /       exactly as it appears in your/
0324 20 64 69 72            FCC   / directory listing./
0337 0D0A                   FDB   CRLF
0339 20 20 20 33            FCC   /   3 - exit program/
034C 0D0A 0D0A              FDB   @RLF,CRLF
0350 4D 6F 72 65            FCC   /More to Come ...../
0367 04                     FCB   EOT
            0368    WELCO2  EQU   *
0368 0D0A 0D0A              FDB   CRLF,CRLF
036C 4E 6F 74 65            FCC   /Note:  a) the user may hit [Return]/
038F 20 69 6E 73            FCC   / instead of typing a/
03A3 0D0A                   FDB   CRLF
03A5 20 20 20 20            FCC   /         drive number.  In this case/
03CA 20 74 68 65            FCC   / the last specified/
03DD 0D0A                   FDB   CRLF
03DF 20 20 20 20            FCC   /         drive number will be used/
0402 20 28 6F 72            FCC   / for the FLEX work/
0414 0D0A                   FDB   CRLF
0416 20 20 20 20            FCC   /         drive in the case no previous/
0438 20 64 72 69            FCC   / drive was given)/
044E 0D0A 0D0A              FDB   CRLF,CRLF
0452 20 20 20 20            FCC   /       b) the user has the option of/
0476 20 72 6F 75            FCC   / routing his output/
0489 0D0A                   FDB   CRLF
048B 20 20 20 20            FCC   /         from selection 1 or 2, to/
04AE 20 74 68 65            FCC   / the printer, terminal,/
04C5 0D0A                   FDB   CRLF
04C7 20 20 20 20            FCC   /          or a file.  If "printer"/
04EA 20 69 73 20            FCC   / is selected and the/
04FE 0D0A                   FDB   CRLF
0500 20 20 20 20            FCC   /          FLEX printer module hasn't/
0525 20 62 65 65            FCC   / been loaded, it/
0535 0D0A                   FDB   CRLF
0537 20 20 20 20            FCC   /          (PRINT.SYS) will/
0552 20 61 75 74            FCC   / automatically be loaded/
056A 0D0A                   FDB   CRLF
056C 20 20 20 20            FCC   /          from drive 0.  If "file"/
058F 20 69 73 20            FCC   / is selected, the user/
05A5 0D0A                   FDB   CRLF
05A7 20 20 20 20            FCC   /          will be prompted for the/
05CA 20 22 74 6F            FCC   / "to" file name.  Type/
05E0 0D0A                   FDB   CRLF
05E2 20 20 20 20            FCC   /          this as a standard FLEX/
0604 20 66 69 6C            FCC   / file specification (eg:/
061C 0D0A                   FDB   CRLF
061E 20 20 20 20            FCC   /          0.XXXX.TXT).  The user/
063F 20 77 69 6C            FCC   / will also be asked/
0652 0D0A                   FDB   CRLF
0654 20 20 20 20            FCC   /          whether the file is/
0672 20 22 74 65            FCC   / "text" or "binary".  If/
068A 0D0A                   FDB   CRLF
068C 20 20 20 20            FCC   /          type "binary" is chosen,/
06AF 20 74 68 65            FCC   / the FLEX space/
06BE 0D0A                   FDB   CRLF
06C0 20 20 20 20            FCC   /          compression flag will be/
06E3 20 73 65 74            FCC   / set and no expansion/
06F8 0D0A                   FDB   CRLF
06FA 20 20 20 20            FCC   /          of tabs, etc. is performed./
0720 20 20 49 66            FCC   /  If ROUTE=P or T,/
0732 0D0A                   FDB   CRLF
0734 20 20 20 20            FCC   /          tabs will be expanded./
0755 0D0A 0D0A              FDB   CRLF,CRLF
0759 45 6E 74 65            FCC   /Enter [Return] to Continue.../
0776 04                     FCB   EOT

            0777    DIRMSG  EQU   *
0777 0D0A                   FDB   CRLF
0779 20 20 20 20            FCC   /      Directory of Drive /
0792                OUTDRV  RMB   1         put drive# here
0793 0D0A                   FDB   CRLF
0795 04                     FCB   EOT

            0796    ENDIR   EQU   *
0796 20 20 20 45            FCC   /   End of Directory Listing. /
07B3 48 69 74 20            FCC   /Hit [Return] to Continue.../
07CE 04                     FCB   EOT

            07CF    ENDCOP  EQU   *
07CF 20 20 20 45            FCC   /   End of Copy. Enter /
07E5 5B 52 65 74            FCC   /[Return] to Continue.../
07FC 04                     FCB   EOT

            07FD    DRVPRT  EQU   *
07FD 20 20 20 45            FCC   "  Enter drive of CP/M disk: "
081A 04                     FCB   EOT

081B                OUTNAM  RMB   CPMNL
0823 2E                     FCC   /./
0824                OUTEXT  RMB   CPMEL
0827 04                     FCB   EOT

                    *
                    * BLKOUT routine
                    *
0828 20 20 20       SP3     FCC   /   /     3 spaces
0828 04                     FCB   EOT

                    *
                    * COPYIT routine
                    *
            082C    INFILE  EQU   *
082C 20 20 20 45            FCC   /   Enter "from" file: /
0842 04                     FCB   EOT

            0843    INDRV   EQU   *
0843 20 20 20 45            FCC   /   Enter "from" drive# /
085A 04                     FCB   EOT
```

```
                    0858 OUTFIL EQU  *
0858 20 20 20 4F           FCC  /   Output to File(F), Printer(P), /
087D 6F 72 20 54           FCC  /or Terminal(Tel)? /
088E 04                    FCB  EOT

                    088F TOFILE EQU  *
088F 20 20 20 45           FCC  /   Enter "to" file: /
0883 04                    FCB  EOT
                    0884 BADFIL EQU  *
0884 46 49 4C 45           FCC  /FILE DOES NOT EXIST!/
0888 04                    FCB  EOT

                    0889 INVSPC EQU  *
0889 49 4E 56 41           FCC  /INVALID FILE SPECIFICATION!/
08D4 04                    FCB  EOT

                    08D5 EXIST  EQU  *
08D5 53 4F 52 52           FCC  /SORRY, FILE ALREADY EXISTS/
08EF 04                    FCB  EOT

                    08F0 FTYPE  EQU  *
08F0 20 20 20 49           FCC  /   Is file type Binary(B) /
090A 6F 72 20 54           FCC  /or Text(Tel)? /
0917 04                    FCB  EOT

                    0918 DTYPE  EQU  *
0918 20 20 20 49           FCC  /   Is this an 8" disk/
0920 20 28 64 65           FCC  : (default is 5") (Y/N)? :
0946 04                    FCB  EOT

                         *
                         * Misc
                         *
0947 3C 3C 3C 20    RDERR  FCC  /<<<< READ ERROR >>>!/
0958 04                    FCB  EOT

0958 50 52 49 4E    PSYS   FCC  /PRINT/
0960 00 00 00              FCB  0,0,0
0963 53 59 53              FCC  /SYS/

                    0966 INTRO  EQU  *
0966 000A 000A             FDB  CRLF,CRLF
096A 20 20 20 46           FCC  *   FLEX/CPM UTILITY (C) 1983*
0986 20 62 79 20           FCC  * by Scott Fraser*
0996 000A 000A             FDB  CRLF,CRLF
099A 20 20 20 20           FCC  /          0 - Directions/
09B2 20 20 20 20           FCC  *          1 - CP/M Directory*
09CE 000A                  FDB  CRLF
09D0 20 20 20 20           FCC  /          2 - Copy File/
09E7 20 20 20 20           FCC  /           3 - EXIT Program/
0A02 000A 000A             FDB  CRLF,CRLF
0A06 20 20 20 45           FCC  /   ENTER SELECTION: /
0A1A 04                    FCB  EOT

                    0A1B BADIN  EQU  *
0A1B 20 20 20 49           FCC  /   INVALID SELECTION, REENTER: /
0A3A 04                    FCB  EOT

0A3B                FILFCB RMB  FCBLEN   reserve FCB area

                         *
                         * Define some CP/M constants
                         *
                    0008 SECBLK EQU  8         8 sectors per block
                    0080 SSIZ8  EQU  128       #bytes per sector (8")
                    0100 SSIZ5  EQU  256       #bytes per sector (5")
                    0002 OFF8   EQU  2         data offset for 8" disk
                    0003 OFF5   EQU  3         data offset for 5" disk
                    001A EOF    EQU  $1A       delimits end of file (CNTL Z)
                    0000 DIRBEG EQU  0         block # of start of CP/M dir
                    00E5 DAREMD EQU  $E5       signifies end of a dir block
                    0009 TABCHR EQU  $09       universal tab char (<cntl>I)

0B7B                DRV    RMB  1         holds current drive#
0B7C 00             OPNFLG FCB  0         file open bits
                    0080 OPN    EQU  %10000000 #file is open
                    0040 TEX    EQU  %01000000 #file is text
                    0020 BINRY  EQU  %00100000 #file is binary
0B7D                TTYWID RMB  1         save TTY Width value here
0B7E 00             CONFIG FCB  0         0=5" disk, otherwise 8" disk

0B7F                BUFSIZ RMB  2         holds buffer size
0B81                BLKCNT RMB  2         size of file is secters
0B83                EOBUF  RMB  2         end of buffer address
0B85                SECSIZ RMB  2         holds sector size
0B87                MAPADR RMB  2         holds addr of map tble
0B89                MAPLEN RMB  1         holds len of map tble

                    0800 BSIZ5  EQU  SSIZ5*SECBLK size of buffer (5")
                    0400 BSIZ8  EQU  SSIZ8*SECBLK size of buffer (8")

0B8A                BUFFER RMB  BSIZ5*2   alloc max buffer size

                    0050 EXTSIZ EQU  CPMBS*5   block list size
;98A                BLKLST RMB  EXTSIZ    reserve for block list


                         *ORG UCA
1B0A 20 01          START  BRA  START1    ***** Begin Program *****

1B0C 04                    FCB  4         <<<< VERSION # >>>>


                    1B0D START1 EQU  *
1B0D 10FE CC2B             LDS  #MEMEND   initialize system stack ptr

1B11 8E 0918               LDX  #DTYPE    ask user for disk type
1B14 BD CD1E               JSR  PSTRNG
1B17 BD CD09               JSR  INCH      get response from user
1B1A 84 5F                 ANDA #$5F      lowercase->uppercase

1B1C 81 59                 CMPA #'Y       reply is yes for 8"
1B1E 26 03                 BNE  NO8       branch if 5" (default)

1B20 7C 0B7E               INC  CONFIG    set config for 8"
                    1B23 NO8  EQU  *
                         *
                         * Initialize some constants first
                         *
1B23 108E 2097             LDY  #MAPTB5   addr of mapping tble
1B27 7D 0B7E               TST  CONFIG    setup buffer size for
1B2A 27 11                 BEQ  FIVE      type of disk

1B2C 8E 0400               LDX  #BSIZ8
1B2F 108E 207D             LDY  #MAPTB8   map tble for 8" disk
1CC3 CC 0080               LDD  #SSIZ8    sector size for 8" disk
1CC6 FD 0B85               STD  SECSIZ    save sector size
1CC9 86 1A                 LDA  #TBLEN8   map tble len for 8" disk
1CCB 20 08                 BRA  STSIZ
                    1C00 FIVE EQU  *
1C00 8E 0800               LDX  #BSIZ5
1C10 CC 0100               LDD  #SSIZ5    sector size
1C13 FD 0B85               STD  SECSIZ    save it
1C16 86 0A                 LDA  #TBLEN5   map tble len for 5" disk
                    1C18 STSIZ EQU  *
1C18 BF 0B7F               STX  BUFSIZ    save size of buffer
1C1B 108F 0B87             STY  MAPADR    save addr of map tble
1C1F B7 0B89               STA  MAPLEN    save len of map tble
1C22 B6 CC0C               LDA  #WDRV     default to work drive
1C25 B7 0B7B               STA  DRV       save default
1C28 8E 1C34               LDX  #MENU
1C2B BF CC16               STX  BADRR     modify escape return register
1C2E B6 BC04               LDA  WIDTH     set current TTY line width
1C31 B7 0B7D               STA  TTYWID    and save it

                    1C34 MENU1 EQU  *
1C34 7F CC1A               CLR  CLN       clear current line num 1st
                    1C37 MENU2 EQU  *
1C37 7C CC22               INC  OSWFCH    set output switch
1C3A 8E 0966               LDX  #INTRO    print intro msg
1C3D BD CD1E               JSR  PSTRNG    ask for selection

                    1C40 PROM  EQU  *
1C40 BD CD09               JSR  INCH      get a response
1C43 BD CD21               JSR  CLASS     classify it
1C46 25 42                 BCS  NO        bad selection
1C48 81 33                 CMPA #'3       over 3?
```

```
1C4A 22 3E          BHI   MX        bad selection

1C4C 84 0F          ANDA  #%00001111 keep low 4 bits only
1C4E 48             ASLA            selection=selection*2
1C4F 8E 1C54        LDX   #JT       addr of jump table
1C52 30 86          LEAX  A,X       addr proper selection
1C54 6E 84          JMP   0,X       goto selection

1C56 20 06      JT  BRA   M0        selection=0
1C58 20 18          BRA   M1        =1
1C5A 20 20          BRA   M2        =2
1C5C 20 29          BRA   M3        =3

            1C5E M0     EQU   *     do selection 0
1C5E 8E 012A        LDX   #WELCOM   print instructions
1C61 BD CD1E        JSR   PSTRNG

1C64 B6 CC03        LDA   DEPTH     breather
1C67 B7 CC1A        STA   CLN
1C6A 8E 0368        LDX   #WELCO2   more to come
1C6D BD CD1E        JSR   PSTRNG
1C70 20 20          BRA   SETPAU

            1C72 M1     EQU   *
1C72 8D 32          BSR   PDIR      print dir
1C74 8E 0796        LDX   #ENDDIR
1C77 BD CD1E        JSR   PSTRNG
1C7A 20 16          BRA   SETPAU

            1C7C M2     EQU   *
1C7C 17 0139        LBSR  COPYIT    copy file
1C7F 8E 07CF        LDX   #ENDCOP
1C82 BD CD1E        JSR   PSTRNG
1C85 20 0B          BRA   SETPAU

            1C87 M3     EQU   *
1C87 7E CD03        JMP   WARMS     return to FLEX

            1C8A MX     EQU   *     bad selection
1C8A 8E 0A1B        LDX   #BADIN
1C8D BD CD1E        JSR   PSTRNG    print msg
1C90 20 AE          BRA   PROM

            1C92 SETPAU EQU   *
1C92 B6 CC03        LDA   DEPTH     pause (if set)
1C95 B7 CC1A        STA   CLN       so user can read stuff
1C98 20 90          BRA   NEXC

            1C9A ERROR2 EQU   *
1C9A BD CD3F        JSR   RPTERR    report error first
1C9D 3E 0947        LDX   #ERDERR   get msg

            1CA0 ERR    EQU   *
1CA0 BD CD1E        JSR   PSTRNG    print msg
1CA3 7E CD03        JMP   WARMS     and return to FLEX


                *
                * Name      - PDIR
                * Function  - This routine is called to print out
                *            directory information on a CP/M disk.
                *            The directory starts at block 0.
                *
                *            No parms are required
                *            All registers are used and not restored
                *
            1CA6 PDIR   EQU   *
1CA6 32 E9 FFFC     LEAS  -LSIZ,S   alloc local storage

1CAA 8E 07FD        LDX   #DRVPMT   ask for a drive
1CAD BD CD1E        JSR   PSTRNG
1CB0 17 01D4        LBSR  GETDRV    get drive #
1CB3 25 78          BCS   PDXIT     if bad then leave

1CB5 17 0427        LBSR  ROUTE     where to route output
1CB8 25 73          BCS   PDXIT     leave if bad
1CBA BD CD24        JSR   PCRLF
1CBD B6 087B        LDA   DRV       get drive
1CC0 B7 CB43        STA   SYSFCB+FCBDN save in FCB
1CC3 8A 30          ORA   #'0       convert to ascii
1CC5 B7 0792        STA   OUTDRV    save to print

1CC8 8E 0777        LDX   #DIRMSG   print msg
1CCB 17 0401        LBSR  PDATA

1CCE 8E 0106        LDX   #HEAD     print dir header
1CD1 17 03FB        LBSR  PDATA

1CD4 FC 087F        LDD   BUFSIZ    get buffer size
1CD7 58             ASLB            and double it to hold
1CD8 49             ROLA            all of dir data
1CD9 8E 088A        LDX   #BUFFER   calc end of buf ptr
1CDC 30 88          LEAX  D,X
1CDE AF E9 0002     STX   DSIZ1,S   and save it

1CE2 8E 088A        LDX   #BUFFER   where to put data
1CE5 86 00          LDA   #DIRSEC   start with 1st block of dir
1CE7 17 02F8        LBSR  GETBLK    get first block
1CEA 4C             INCA
1CEB 17 02F4        LBSR  GETBLK    and get second block
                *
                * BUFFER now has all directory data in it
                *
1CEE 8E 088A        LDX   #BUFFER   position to 1st entry

            1CF1 PD1    EQU   *
1CF1 AF E9 0000     STX   ENTADR,S  save entry addr

1CF5 A6 84          LDA   CPMFEF,X  set entry indicator
1CF7 81 E5          CMPA  #DIREND   does this entry exist?
1CF9 26 05          BNE   PD7       branch if yes
1CFB 30 88 20       LEAX  CPMLEN,X  goto next entry
1CFE 20 27          BRA   PD8       and see if done yet

                *
                * Now print file name
                *
            1D00 PD7    EQU   *
1D00 30 01          LEAX  CPMNAM,X  "from" field
1D02 C6 08          LDB   #CPMEL    length to move
            1D04 PD5    EQU   *
1D04 A6 80          LDA   0,X+      get a char
1D06 26 02          BNE   *+2+2     skip if not zero
1D08 86 20          LDA   #SP
1D0A BD CD18        JSR   PUTCHR    print char
1D0D 5A             DECB
1D0E 27 08          BEQ   PD6       exit if done
1D10 C1 03          CMPB  #CPMEL    at extension yet?
1D12 26 F0          BNE   PD5       branch if not
1D14 86 2E          LDA   #'.       print a '.' first
1D16 BD CD18        JSR   PUTCHR
1D19 20 E9          BRA   PD5       and continue

            1D1B PD6    EQU   *
1D1B AE E9 0000     LDX   ENTADR,S
1D1F 17 00F4        LBSR  EXTNTS    get size of file and
1D22 BD 25          BSR   BLKOUT    print size and list
1D24 BD CD24        JSR   PCRLF

            1D27 PD8    EQU   *
1D27 AC E9 0002     CMPX  DSIZ1,S   end of buffer yet?
1D2B 25 C4          BLO   PD1       branch if not
            1D2D PDXIT  EQU   *
1D2D 17 0485        LBSR  RESET     clear things up first

1D30 32 E9 0004     LEAS  LS1,S     release local storage
1D34 39             RTS             return

                *
                * Data area for PDIR routine
                *
            1D35 LPC    SET   *
0090                ORG   $0090
            0000 LCL    SET   0
0000        ENTADR  RMB   2
0002        DSIZ1   RMB   2
            0004 LS1    EQU   *-LCL     len of local area

1D35                ORG   LPC       restore PC

                *
```

```
                    * Name      - UNKOUT
                    * Function - This routine prints out the 4 byte
                    *            field in the CPM dsect starting with
                    *            field "CPMCF". Printed only for
                    *            debugging purposes.
                    *
                    *            On entry the X reg points to the
                    *            beginning of the 4 byte field.
                    *
                    *            Regs A and B are preserved.
                    *
               1D35 UNKOUT EQU  *
1D35 34 06          PSHS   D          save regs
1D37 86 20          LDA    #SP        print a space first
1D39 BD CD18        JSR    PUTCHR

1D3C C6 04          LDB    #4         print 4 bytes

               1D3E UNK1   EQU  *
1D3E BD CD3C        JSR    OUTHEX     print in hex
1D41 30 01          LEAX   1,X        to next byte
1D43 5A             DECB              done yet?
1D44 26 F8          BNE    UNK1       branch if not

1D46 35 06          PULS   D          restore regs
1D48 39             RTS               and return


                    *
                    * Name      - BLKOUT
                    * Function - This routine prints out the size of
                    *            the current file entry. ACC A contains
                    *            the size of the file, and the area
                    *            BUKLST contains the list of blocks

                    *            belonging to the current file.
                    *
                    *            All registers are preserved
                    *
               1D49 BLKOUT EQU  *
1D49 34 36          PSHS   A,B,X,Y    save regs
1D4B 32 E9 FFF0     LEAS   -LS2,S     alloc local storage

1D4F 1F 89          TFR    A,B        copy file size
1D51 86 20          LDA    #SP        print a space
1D53 BD CD18        JSR    PUTCHR

1D56 4F             CLRA
1D57 7D 0B7E        TST    CONFIG     is the 5" or 8" disk?
1D5A 26 01          BNE    *+2+1      do not double if 8"
1D5C 58             ASLB              double file size for 5" disk
1D5D ED E9 0000     STD    OUTSIZ,S   save size of file
1D61 30 E9 0000     LEAX   OUTSIZ,S   and print it
1D65 C6 01          LDB    #1         set spc suppression flag
1D67 BD CD39        JSR    OUTDEC

1D6A 86 4B          LDA    #'K        with "K" prefix
1D6C BD CD18        JSR    PUTCHR

1D6F 8E 0828        LDX    #SP3       print spaces
1D72 17 035A        LBSR   PDATA

1D75 8E 0881        LDX    #BLKCNT    print file size in sectors
1D78 C6 01          LDB    #1         set spc suppression flag
1D7A BD CD39        JSR    OUTDEC
1D7D 8E 0828        LDX    #SP3
1D80 17 034C        LBSR   PDATA

1D83 8E 1D8A        LDX    #BLKLST    point to block list
1D86 F6 CC29        LDB    COC        save current col #
1D89 E7 E9 0002     STB    COLUME,S
               1D8D BLK5   EQU  *
1D8D C6 10          LDB    #16        print 16 blks/line

               1D8F BLK3   EQU  *
1D8F 6D 84          TST    0,X        end of list yet?
1D91 27 1E          BEQ    BLK4       branch if yes

1D93 BD CD3C        JSR    OUTHEX     print block#
1D96 86 20          LDA    #SP        print space
1D98 BD CD18        JSR    PUTCHR

1D9B 30 01          LEAX   1,X        go to nxt block
1D9D 5A             DECB              printed 16 yet?
1D9E 26 EF          BNE    BLK3       branch if not

1DA0 BD CD24        JSR    PCRLF
1DA3 E6 E9 0002     LDB    COLUME,S   space over to col COLUME
               1DA7 BLK6   EQU  *
1DA7 86 20          LDA    #SP
1DA9 BD CD18        JSR    PUTCHR     print space
1DAC 5A             DECB
1DAD 26 F8          BNE    BLK6       continue spacing

1DAF 20 DC          BRA    BLK5       print another line now

               1DB1 BLK4   EQU  *

1DB1 32 E9 0003     LEAS   LS2,S      release local storage
1DB5 35 36          PULS   A,B,X,Y    restore regs
1DB7 39             RTS               return
                    *
                    * Data area for BLKOUT routine
                    *
               1DB8 LPC    SET  *
0000                ORG    $0000
               0000 LCL    SET  *
0000           OUTSIZ RMB  2
0002           COLUME RMB  1
               0003 LS2    EQU  *-LCL    len of local area

1DB8                ORG    LPC        restore PC
                    *
                    * Name      - COPYIT
                    * Function - This rtn is called to copy a CP/M file
                    *            to a FLEX file. The user is prompted
                    *            for a "from" file (off the CP/M disk)
                    *            and a "to" file (in FLEX). If the "to"
                    *            file is not specified, output goes to
                    *            the terminal.
                    *
                    *            All registers are used and not restored
                    *
               1DB8 COPYIT EQU  *
1DB8 32 E9 FFF5     LEAS   -LS3,S     alloc local storage

1DBC 8E 0843        LDX    #INDRV     get "from" drive
1DBF BD CD1E        JSR    PSTRNG
1DC2 17 00C2        LBSR   GETDRV     get drive#
1DC5 25 47          BCS    DP2        if bad then leave

1DC7 8E 082C        LDX    #INFILE    prompt for "from" filename
1DCA BD CD1E        JSR    PSTRNG
1DCD BD CD18        JSR    INBUF      get "from" filename
1DD0 30 E9 0000     LEAX   SVEFIL,S   where to put filename
1DD4 17 0005        LBSR   GTSPEC     get CP/M filename
1DD7 25 35          BCS    DP2        if bad then leave

1DD9 17 0110        LBSR   SRCH       go do search
1DDC 24 08          BCC    GETO       if found, get "to" file
                    *
                    * File not found in CP/M directory -> flag
                    * and reprompt user.
                    *
1DDE 8E 0884        LDX    #BADFIL    print msg
1DE1 BD CD1E        JSR    PSTRNG
1DE4 20 28          BRA    DP2        return

                    *
               1DE6 GETO   EQU  *
                    *
                    * The CP/M directory entry has been found.
                    * This entry and possibly subsequent entries,
                    * contains a list of block extents that the
                    * file covers on the disk. Copy all these
area
                    *
1DE6 8D 2E          BSR    EXTNTS     set all the extents
                    *
                    * Find out where to route the output
                    *
1DE8 17 02F4        LBSR   ROUTE      set route
```

```
1DEB 25 21              BCS   DP2       leave if bad

1DED B6 0B7B            LDA   DRV       get "from" drive #
1DF0 B7 CB43            STA   SYSFCB+FCBDN and save in FCB
1DF3 108E 1B0A          LDY   #BUFLST   Y->list of file blocks
1DF7 7D 0B7C            TST   OPNFLG    writing to a file?
1DFA 26 03              BNE   DP1       yes, then no lead CRLF

1DFC BD CD24            JSR   PCRLF     print CRLF first

                1DFF DP1  EQU   *
1DFF A6 A0              LDA   0,Y+      get a block
1E01 27 0B              BEQ   DP2       until block# = X'00'

1E03 8E 0B0A            LDX   #BUFFER
1E06 17 D109            LBSR  GETBLK
1E09 17 0134            LBSR  PRTBLK    dump to output device
1E0C 20 F1              BRA   DP1       continue dump

                1E0E DP2  EQU   *
1E0E 17 03A4            LBSR  RESET     clear things up first

1E11 32 E9 0008         LEAS  LS3,S     release local storage
1E15 39                 RTS             return
                        *
                        * Data area for COPYIT routine
                        *
                1E16 LPC  SET   *
0000                    ORG   $0000
                0000 LCL  SET   *
0000            SVEFIL  RMB   CPMNL
                000B LS3  EQU   *-LCL   len of local storage

1E16                    ORG   LPC       restore PC
                        * Name     - EXTNTS
                        * Function - This routine copies a list of file blocks
                        *            to a save area. On entry:
                        *
                        *            X -> a CP/M file entry
                        *
                        *            The blocks belonging to a file may span
                        *            more than one file entry. If the list
                        *            does, the following file entry(s) will
                        *            contain a X'01' in the first byte of
                        *            the CPMEF field. All blocks are saved
                        *            in the BUFLST area!
                        *
                        *            On exit, Y is unchanged and X points to
                        *            the file entry immediately following the
                        *            entry(s) belonging to the file in
                        *            question.
                        *            ACC A contains the number of blocks
                        *            transferred.
                        *
                        *            ACC B is unchanged
                        *
                1E16 EXTNTS EQU   *
1E16 34 24              PSHS  B,Y       save regs
1E18 32 E9 FFFD         LEAS  -LS4,S    alloc local storage

1E1C 6F E9 0000         CLR   BCNT,S    start ctr at 0
1E20 AF E9 0001         STX   XTMP,S    save X reg
1E24 7F 1B0A            CLR   BUFLST    zero out block area
1E27 8E 1B0A            LDX   #BUFLST   "from" field
1E2A 31 01              LEAY  1,X       "to" field
1E2C CC 004F            LDD   #EXTSIZ-1 len to move
1E2F 17 0279            LBSR  MVC       move in X'00's

1E32 AE E9 0001         LDX   XTMP,S    restore file ent ptr
1E36 108E 1B0A          LDY   #BUFLST   where to put blocks
1E3A 7F 0B81            CLR   BLKCNT
1E3D 7F 0B82            CLR   BLKCNT+1  start sector counter

                1E40 EX1  EQU   *
1E40 4F                 CLRA            accumulate file/sector count
1E41 E6 0F              LDB   CPMSC,X   get file/sector count
1E43 F3 0B81            ADDD  BLKCNT
1E46 FD 0B81            STD   BLKCNT    and save

1E49 86 10              LDA   #CPMBS    move at most CPMBS bytes at a time
1E4B AF E9 0001         STX   XTMP,S    save file entry ptr

1E4F 30 88 10           LEAX  CPMBLK,X  get to file's blklist
                1E52 EX2  EQU   *
1E52 E6 80              LDB   0,X+      get a block
1E54 27 1D              BEQ   EX3       if zero then done
1E56 E7 A0              STB   0,Y+      save block#
1E58 6C E9 0000         INC   BCNT,S    and incr block counter
1E5C 4A                 DECA            done yet?
1E5D 26 F3              BNE   EX2       branch if not

1E5F AE E9 0001         LDX   XTMP,S    restore ptr to file entry
1E63 30 88 2C           LEAX  CPMLEN+CPMEF,X ptr to next file's
1E66 6D 84              TST   0,X       continuation field
1E68 27 09              BEQ   EX3       and exit if X'00'

1E6A AE E9 0001         LDX   XTMP,S    file is continued
1E6E 30 88 20           LEAX  CPMLEN,X  so goto next file
1E71 20 CD              BRA   EX1       and get it's block list

                1E73 EX3  EQU   *
1E73 6F A4              CLR   0,Y       save trailer of X'00'
1E75 AE E9 0001         LDX   XTMP,S    leave X pointing to next
1E79 30 88 20           LEAX  CPMLEN,X  NEW file entry
1E7C A6 E9 0000         LDA   BCNT,S    get block counter

1E80 32 E9 0003         LEAS  LS4,S     release local storage
1E84 35 24              PULS  B,Y       restore regs
1E86 39                 RTS             and return
                        *
                1E87 LPC  SET   *
0000                    ORG   $0000     define local stor dsect
                0000 LCL  SET   *
0000            BCNT    RMB   1
0001            XTMP    RMB   2
                0003 LS4  EQU   *-LCL   len of local storage

1E87                    ORG   LPC       restore PC

                        *
                        * Name     - GETDRV
                        * Function - This routine gets a drive # from the
                        *            terminal. The user can either enter a
                        *            drive # in the range 0-3, or type null,
                        *            in which case the default drive# in
                        *            "DRV" will be taken.
                        *
                        *            The carry is clear if # ok, else it is
                        *            set. The valid drive number is returned
                        *            in ACC A, and is also saved in DRV.
                        *
                        *            All registers are preserved.
                        *
                1E87 GETDRV EQU   *
1E87 34 34              PSHS  B,X,Y     save regs

1E89 BD CD09            JSR   INCH      get response
1E8C 81 0D              CMPA  #CR       use default?
1E8E 26 05              BNE   GET1      branch if not

1E90 B6 0B7B            LDA   DRV       get default

1E93 20 0E              BRA   GET4      and save it

                1E95 GET1 EQU   *
1E95 BD CD21            JSR   CLASS     classify it
1E98 25 00              BCS   GET2      bad drive
1E9A 81 33              CMPA  #'3       in range 0-3?
1E9C 22 09              BHI   GET2      bad drive

1E9E 84 0F              ANDA  #%00001111 keep low 4 bits
1EA0 B7 0B7B            STA   DRV       save in DRV storage
                1EA3 GET4 EQU   *
1EA3 1C FE              CLC             set good return
1EA5 20 02              BRA   GET3      and return

                1EA7 GET2 EQU   *
1EA7 1A 01              SEC             set bad return
                1EA9 GET3 EQU   *
1EA9 35 34              PULS  B,X,Y     restore regs
1EAB 39                 RTS             and return
```

```
                    *
                    * Name     - GTSPEC
                    * Function - This routine parses the system INBUF
                    *            buffer for a CP/M file name. A valid
                    *            filename must contain up to 8 chars,
                    *            then an optional '.' and up to 3 char
                    *            file extension. The X reg points to
                    *            an area to place the 11 char file name
                    *            and any unused chars are padded with
                    *            spaces.
                    *
                    *            Carry is clear if file spec ok, else
                    *            it is set.
                    *
                    *            All regs are preserved.
                    *
          1EAC      GTSPEC  EQU   *
1EAC 34   36                PSHS  A,B,X,Y     save regs

                    *
                    * First blank out the "to" field
                    *
1EAE 86   20                LDA   #SP         pad with spaces
1EB0 A7   84                STA   0,X
1EB2 31   01                LEAY  1,X         "to" field
1EB4 CC   000A              LDD   #CPMFNL-1   length to move
1EB7 17   01F1              LBSR  MVC         move in spaces

1EBA 10BE C080              LDY   #LINEBUF    pt to system buffer
1EBE 86   08                LDA   #CPMFNL     8 chars to move

1EC0 E6   A4                LDB   0,Y         user just hit return?
1EC2 C1   00                CMPB  #CR
1EC4 27   21                BEQ   GTBAD       yes, then bad return
          1EC6      GT1     EQU   *
1EC6 E6   A0                LDB   0,Y+        get a char

1EC8 C1   20                CMPB  #SP         space here?
1ECA 27   FA                BEQ   GT1         skip spaces

1ECC C1   2E                CMPB  #'.         have an extension
1ECE 26   0A                BNE   GT2          nope, then continue
                    *
                    * The rest of the filename from the system
                    * buffer is the extension. Adjust the
                    * "to" ptr so that any unused chars in the
                    * "to" name are spaces
                    *
1ED0 1F   89                TFR   A,B         copy A
1ED2 C0   03                SUBB  #CPMEL
1ED4 2B   11                BMI   GTBAD       if neg then bad name
1ED6 30   85                LEAX  B,X         adjust "to" ptr
1ED8 20   EC                BRA   GT1         and continue

          1EDA      GT2     EQU   *
1EDA C1   00                CMPB  #CR         end yet?
1EDC 27   05                BEQ   GT3         done if so

1EDE E7   80                STB   0,X+        save to "to" field
1EE0 4A                     DECA              moved all chars?
1EE1 26   E3                BNE   GT1         branch if not

          1EE3      GT3     EQU   *
1EE3 1C   FE                CLC               good return
1EE5 20   02                BRA   GTGOOD
          1EE7      GTBAD   EQU   *
1EE7 1A   01                SEC               set bad return
1EE9 35   36        GTGOOD  PULS  A,B,X,Y     restore regs
1EEB 39                     RTS               and return


                    *
                    * Name     - SRCHM
                    * Function - This routine searches the directory on
                    *            a CP/M disk for the file name pointed to
                    *            by the X register. "SYSFCB" is the FCB
                    *            used, and is assumed to contain the
                    *            drive where the CP/M disk lies
                    *
                    *            On exit, X->CP/M file entry of found
                    *            file. If file is not found, the carry
                    *            is set, otherwise it is clear
                    *
                    *            All registers are preserved
                    *
          1EEC      SRCHM   EQU   *
1EEC 34   26                PSHS  A,B,Y       save regs
1EEE 32   E9 FFFE           LEAS  -LSS,S      alloc local storage

1EF2 1F   12                TFR   X,Y         copy file ptr to Y
1EF4 FC   087F              LDD   BUFSIZ      get size of buffer
1EF7 58                     ASLB              and double to hold all
1EF8 49                     ROLA              dir data
1EF9 8E   088A              LDX   #BUFFER     calc end of buf ptr
1EFC 30   88                LEAX  D,X
1EFE AF   E9 0000           STX   DSIZ2,S     and save it

1F02 8E   088A              LDX   #BUFFER     where to put dir data
1F05 86   00                LDA   #DIRBEG     get first dir block
1F07 17   0008              LBSR  GETBLK      and read it in
1F0A 4C                     INCA              get next block
1F0B 17   0004              LBSR  GETBLK      and read it in too
1F0E 1F   21                TFR   Y,X         copy back file ptr to X

1F10 108E 088A              LDY   #BUFFER     Y-> dir data
          1F14      SRC2    EQU   *
1F14 A6   A4                LDA   CPMFEF,Y    file entry exists?
1F16 81   E5                CMPA  #DIRMAX
1F18 27   0D                BEQ   SRC3        branch if not
                    *
                    * Compare dir file with file set down to
                    * see if a match
                    *
1F1A 34   20                PSHS  Y           save entry ptr
1F1C 31   21                LEAY  CPMFCB,Y    Y->string2
1F1E 86   0B                LDA   #CPMFNL     len to compare
1F20 17   019E              LBSR  CLC         compare

1F23 35   20                PULS  Y           restore entry ptr
1F25 27   0E                BEQ   FOUND       branch if a match

          1F27      SRC3    EQU   *
1F27 31   A8 20             LEAY  CPMLEN,Y    go to next entry
1F2A 10AC E9 0000           CMPY  DSIZ2,S     finished?
1F2F 25   E3                BLO   SRC2        branch if not

1F31 1A   01                SEC               file not found
1F33 20   04                BRA   RET         return

          1F35      FOUND   EQU   *
1F35 1C   FE                CLC               file found
1F37 1F   21                TFR   Y,X         X-> found entry
          1F39      RET     EQU   *

1F39 32   E9 0002           LEAS  LSS,S       release local storage
1F3D 35   26                PULS  A,B,Y       restore regs
1F3F 39                     RTS               and return
                    *
                    * Data area for SRCHM routine
                    *
          1F40      LPC     SET   *
0000                        ORG   #0000
          0000      LCL     SET   *
0000                DSIZ2   RMB   2
          0002      LSS     EQU   *-LCL       len of local storage

1F40                        ORG   LPC         restore PC


                    *
                    * Name     - PRTBLK
                    * Function - This routine dumps the contents of
                    *            BUFFER to the standard output.
                    *
                    *            All registers are preserved
                    *
                    *
          1F40      PRTBLK  EQU   *
1F40 34   36                PSHS  A,B,X,Y     save regs
```

```
1F42 FC 0881          LDD   BLKCNT    get sector counter
1F45 1083 0008        CMPD  #8        in last blk yet?
1F49 22  34           BHI   PB6       branch if not

1F4B 8E  088A         LDX   #BUFFER   pt to beginning of buf
             1F4E PB7 EQU   *
1F4E 7D  087E         TST   DDNFIG    is this a 5" disk
1F51 27  06           BEQ   PB13      branch if so
1F53 30  89 0080      LEAX  SSIZ8,X   this is an 8"
1F57 20  04           BRA   PB14
             1F59 PB13 EQU  *
1F59 30  89 0100      LEAX  SSIZ5,X   incr by 5" size
             1F5D PB14 EQU  *
1F5D 5A               DECB
1F5E 26  EE           BNE   PB7

                      *
                      * The block reached is the EOF block.
                      * Scan from the end of this block, back
                      * until a non-EOF character is reached
                      *
1F60 B6  087C         LDA   OPNFLG    get open flag
             1F63 PB9 EQU   *
1F63 E6  1F           LDB   -1,X      get a char
1F65 C1  E5           CMPB  #DIREND   got an EOF char?
1F67 27  0D           BEQ   PB11      branch if so

1F69 81  C0           CMPA  #OPN+TEX  is this a text file?
1F6B 27  05           BEQ   PB10      then also check another EOF char

1F6D 7D  087C         TST   OPNFLG    maybe we have no o/p file?
1F70 26  08           BNE   PB12      set EOF addr if so

             1F72 PB10 EQU  *
1F72 C1  1A           CMPB  #EOF      check if this EOF char

1F74 26  04           BNE   PB12      set EOF addr if not

             1F76 PB11 EQU  *
1F76 30  1F           LEAX  -1,X      go back a character
1F78 20  E9           BRA   PB9       and continue checking for EOF

             1F7A PB12 EQU  *
1F7A BF  0883         STX   EOBUF     set EOF address here
1F7D 20  11           BRA   PB8       and leave

             1F7F PB6  EQU  *
1F7F 83  0008         SUBD  #8        reduce sector counter
1F82 FD  0881         STD   BLKCNT    and save back
1F85 FC  087F         LDD   BUFSIZ    set end of buf addr to default
1F88 8E  088A         LDX   #BUFFER
1F8B 30  88           LEAX  D,X
1F8D BF  0883         STX   EOBUF

             1F90 PB8  EQU  *
1F90 8E  088A         LDX   #BUFFER   point to buffer area
1F93 5F               CLRB            start column counter at 0

             1F94 PB1  EQU  *
1F94 BC  0883         CMPX  EOBUF     end of buffer yet?
1F97 27  46           BEQ   PB3       branch if so
1F99 A6  80           LDA   0,X+      get a char

1F9B 34  04           PSHS  B         save col ctr
1F9D F6  087C         LDB   OPNFLG    get open flag
1FA0 C1  A0           CMPB  #OPN+BINRY file open and binary?
1FA2 35  04           PULS  B         restore ctr
1FA4 27  33           BEQ   PB4       branch if so

1FA6 81  0A           CMPA  #LF       line feed?
1FA8 27  EA           BEQ   PB1       skip if so

1FAA 81  0D           CMPA  #CR       a carriage return?
1FAC 26  08           BNE   PB2       branch if not

1FAE 5F               CLRB            clear column counter
1FAF 7D  087C         TST   OPNFLG    writing to a file?
1FB2 26  25           BNE   PBA       branch if so
1FB4 BD  CD24         JSR   PRCRLF     print a CRLF
1FB7 20  DB           BRA   PB1
             1FB9 PB2  EQU  *
```

```
1FB9 81  09           CMPA  #TABCHR   a tab char?
1FBB 26  1C           BNE   FB4       nope, then output char as is

                      *
                      * Encountered a tab character. If outputting
                      * to a file, the tab must be converted to the
                      * proper spaces, depending on the current
                      * column.
                      *
1FBD 34  04           PSHS  B         save current column ctr
1FBF C4  07           ANDB  #%00000111 B = mod(curcol,8)
1FC1 86  08           LDA   #8
1FC3 34  04           PSHS  B
1FC5 A0  E0           SUBA  0,S+      A = 8 - mod(curcol,8)
1FC7 34  02           PSHS  A         save this value too
1FC9 1F  89           TFR   A,B       B = #spaces to insert

1FCB B6  20           LDA   #SP
             1FCD PB5 EQU   *
1FCD BD  CD1B         JSR   PUTCHR    output a space
1FD0 5A               DECB            until all done
1FD1 26  FA           BNE   PB5

1FD3 35  04           PULS  B         set back # spaces
1FD5 EB  E0           ADDB  0,S+      add to cur column ctr
1FD7 20  B8           BRA   PB1       and continue

             1FD9 PB4  EQU  *
1FD9 BD  CD1B         JSR   PUTCHR    output a char as is
1FDC 5C               INCB            incr col ctr
1FDD 20  B5           BRA   PB1       continue
             1FDF PB3  EQU  *
1FDF 35  36           PULS  A,B,X,Y   restore regs
1FE1 39               RTS             and return

                      *
                      * Name     - GETBLK
                      * Function - This routine takes in a CP/M block
                      *           number in ACC A, and reads that block
                      *           into the buffer pointed to by X.
                      *           A CP/M block has SBCBLK
                      *           sectors/block, each sector having SECSIZ
                      *           bytes of information.
                      *
                      *           On exit X points 1 byte past the
                      *           end of the buffer. All other
                      *           registers are preserved.
                      *
             1FE2 GETBLK EQU  *
1FE2 34  26           PSHS  Y,D       save registers
1FE4 32  E9 FFFD      LEAS  -LS6,S    alloc local storage

1FE8 1F  12           TFR   X,Y       copy buf ptr to Y

1FEA 1F  89           TFR   A,B       put blk# in B
1FEC 4F               CLRA            A=0 means read the block
1FED 1F  01           TFR   D,X       X=block to read
1FEF BD  38           BSR   MAPPER    read the block

1FF1 C6  08           LDB   #SECBLK   #sector per block
1FF3 E7  E9 0000      STB   TMP1,S    save it
             1FF7 GB1 EQU   *
1FF7 1F  10           TFR   X,D       save trk/sec value
1FF9 ED  E9 0001      STD   TMP3,S
1FFD 8E  C340         LDX   #SYSFCB   point to an FCB
2000 17  009E         LBSR  READSS    read a sector
2003 10C6 FC93        LBNE  ERR02     branch if error
*
                      *
                      * Transfer bytes just read, to BUFFER
                      *
2007 30  88 40        LEAX  FCBSB,X   "from" field
200A FC  0885         LDD   SBCSIZ    length to move
200D 17  0098         LBSR  MVC       perform move
2010 31  A9           LEAY  D,Y       incr buf addr
```

TO BE CONTINUED

MODEM 68

***** MODEM68 ****

DISTRIBUTION RIGHTS:

I allow unrestricted distribution of this software on a non-profit basis. However any commercial distribution or application using either the programs or documentation requires my written permission in advance.

MODEM68 provides for file transfer between two computers using the FLEX operating system, or between a FLEX based machine and a CP/M computer. FLEX utility commands are supported from within MODEM68. The program uses Ward Christensen's protocol for file transfers with Chuck Forsberg's YAM batch file transfer mode.

The Christensen protocol is widely used in micro-computer applications and has been well documented in Byte and other journals. The only difference for a header block under YAM and a normal Christensen block is that the header has block number 0. The filename is sent as it appears on the screen in CP/M format and may or may not have a period and extension. The filename is terminated by a null. On completion of file transfer(s) a null filen e is sent to terminate the session. For FLEX users, the standard FLEX conventions apply - e.g. .ASM will transfer any files with a ASM extension, however due to differences in filename conventions some CP/M filename characters may be translated into an alpha character for FLEX (lowercase z in this version). Also, any CP/M files received without an extension will default to .TXT.

Files of any size can be handled, as there is no requirement for them to fit into memory. The I/O buffer has been restricted to the Christensen protocol block size of 128 bytes for ease of implementation. In practice this does not significantly increase file transfer times, although it may slow things down in terminal mode with data logging.

USAGE:

The program is menu driven and prompts for all input when required. To return to the main menu, (aborting a file transfer) type a ^E (control E) from the keyboard. All FLEX utilities obey the same conventions they use normally - e.g. COPY 0 1 will copy all files from drive 0 to drive 1.

Switches have been included to provide for batch file transmission, a means of logging all screen display to disk using XON/XOFF protocol and also for echoplex operation. These switches can be viewed by use of the menu item Z, and may be altered if desired. A carriage return entered in response to the prompt will return to the main menu without displaying the remainder of the switch settings. An additional switch to allow for cyclic redundancy checking instead of the normal checksum has been included in the code but not implemented in the current version. This switch must NOT be set.

DESCRIPTION:

The entire pro ram is written in M6800 assembler code to allow for assembly on FLEX2 and FLEX9 computers. Cyclic redundancy file checking has not been implemented in this version, but hooks have been provided in the code to facilitate its inclusion.

MODEM68.TXT

This is a header file which calls in a number of library files - MOD68-1 to MOD68-11.

MOD68-1.TXT

This file contains all equates used. It also contains some buffer space on page 0 for common variables. The soft switches are all defaulted to the OFF state - change these values to $ff if you want them ON.

MOD68-2.TXT

This is the mainline routine. It prints an initial signon message followed by a menu. All other modes (except return to FLEX) are called as subroutines and return to the menu on completion or after entry of ^X from the keyboard. Responses to the menu prompts are independent of case.

MOD68-3.TXT

All subroutines used by more than one module are contained in this file. There is also an initialisation routine for a 6850 ACIA used as a modem port.

MOD68-4.TXT

Resets the stack pointer and returns to FLEX. This routine also calls the file management system to close any files left open.

MOD68-5.TXT

This file contains all file reception stuff. It firstly checks to see what mode is used (normal or batch) and then uses the appropriate routine. Normal mode is fairly straightforward, it prompts for a filename, opens the file and then waits until the line is free to send an initial NAK. Batch mode is mostly concerned with the differences between FLEX and CP/M conventions. Once it gets that all sorted out it calls the normal receive routines until all transfers are completed. You may wish to alter the default compatibility character 'z', if so you will need to patch subroutines first and next.

MOD68-6.TXT

This routine handles terminal communications mode. If in log mode, you will be prompted for a filename to which all screen data will be logged until you return to the main menu. This mode uses XON/XOFF protocol and is useful for capturing data from bulletin boards etc. When the buffer is nearly full, the system sends an XOFF to halt output from the remote computer. It then waits for a bit to see if anything else arrives before it writes the buffer out to disk. After completion of the write, it sends an XON to restart output from the remote computer. At present, the number of characters positions left in the buffer before an XOFF is sent is 5. This seems to work ok with an RCPM running at 300bps but may need adjustment for other systems or speeds. If this is the case, change bufend-5 in build.

If the ECHO switch is set, all input from the keyboard is displayed on screen after being sent to line, and anything received from line is returned. Without echo, only characters received from line are displayed. For dialogue between two systems using MODEM68, one must be in echo mode. Also when communicating with a CP/M computer running MODEM or YAM, the CP/M machine must be in CHAT mode or the FLEX machine must be in echo mode.

MOD68-7.TXT

This file contains the file transmission bits. Similarly to receipt, the mode is checked first. Much of MOD68-5 (receive file) and MOD68-7 (transmit file) are concerned with the differences between FLEX and CP/M conventions. FLEX allows alpha characters only as the first character in a filename and an extension. It also requires an extension, so the additional characters allowed by CP/M are forced to an alpha character when in batch mode. No attempt has been made to implement file transmission times because FLEX uses space compression, so the amount of data stored is less than that transmitted. The batch send routine uses FLEX filename matching conventions. The code is based on the TSC pdel utility, and performs similarly for name matching, however the filename is accepted from within the program instead of appearing on the command line.

MOD68-8.TXT

This routine allows any utility running in the utility command space within FLEX to be run. It issues the normal FLEX prompt when called, and appears to the user as if he is in FLEX. Any program that runs in low memory must not be executed with this routine - e.g. EDIT. When the utility has completed, it will return to the MODEM68 main menu.

MOD68-9.TXT

This routine can be used to abort file transfers from a remote computer.

MOD68-10.TXT

This is a routine to display and/or alter switches. Any additional switches should appear in MOD68-1 before endsw and should have a five letter name (including spaces) included in swmsgf in MOD68-11.

MOD68-11

This file contains the command table, all strings and messages and file control blocks and buffers

ADAPTING TO YOUR SYSTEM:

If using a FLEX2 system with 6850 ACIAs for keyboard and modem ports, the only changes required are to patch the addresses of your modem into aciac and keyboard into kbd in MOD68-1. For FLEX9 users, the base equate in MOD68-1 will also need to be changed.

If you are not using 6850s you will need to supply the appropriate initialisation, status and read and write routines in MOD68-3.

IN CASE OF TROUBLE:

I welcome any suggestions or bug reports. Also any enhancements (such as CRC checking) are encouraged.

John Moorfoot
2 Yarrayne St
Leopold
Victoria 3221
Australia

N.B. FLEX is a trademark of Technical Systems Consultants. CP/M is a trademark of Digital Research.

```
*****************************************************
*                    MODEM68                        *
*                                                   *
*                 By John Moorfoot                  *
*                 2 Yarrayne Street                 *
*                 Leopold                           *
*                 Victoria                          *
*                 Australia 3226                    *
*                                                   *
* Version 1.0   31 August 1983                      *
*                                                   *
* Uses Christensen protocol for file transfers      *
* allows for interchange of text and binary         *
* files between FLEX & CP/M based systems.          *
*                                                   *
* The program is menu driven and provides for       *
* full 8 bit bathways file transfers as well as     *
* a dialog mode (with capture to file). Disk        *
* utilities can be executed from within the         *
* program. Files accomodated are not limited by     *
* memory size. There are switches provided to       *
* enable batch mode transfers (obeys Chuck          *
* Forsberg's YAM conventions); log screen dialog    *
* to disk file; and also to enable local echo -     *
* required on one machine only in dialog mode.      *
*                                                   *
* Entering ^E (control E) at any time from the      *
* keyboard exits the current mode and returns to    *
* the main menu.                                    *
*                                                   *
* FLEX is a trademark of Technical Systems          *
* Consultants.                                      *
* CP/M is a trademark of Digital Research.          *
*****************************************************

* EQUATES *

        *base equ $a000 FLEX2
E000    base    equ     $e000   FLEX9
C002    eol     equ     base+$c82
BCBC    workdr  equ     base+$cbc
CC11    lsttrm  equ     base+$cc11
CC14    bufpnt  equ     base+$cc14
C003    warm    equ     base+$d03
CD15    getch   equ     base+$d15
CD1B    putchr  equ     base+$d1b
CD18    inbuf   equ     base+$d18
CD1E    pstrng  equ     base+$d1e
CD21    class   equ     base+$d21
CD24    pcrlf   equ     base+$d24
CD27    nxtch   equ     base+$d27
CD2D    getfil  equ     base+$d2d
CD33    setext  equ     base+$d33
CD3C    outdec  equ     base+$d3c
CD3F    rpterr  equ     base+$d3f
CD40    docmnd  equ     base+$d40
0403    fmscls  equ     base+$1403
0406    fms     equ     base+$1406

E004    aciac   equ     %1004   modem port
E005    aciad   equ     aciac+1
E008    kbd     equ     $e008   keyboard acia status reg

0000    null    equ     0
0015    nak     equ     $15
0018    can     equ     $18
0006    ack     equ     6
001A    sub     equ     $1a
0004    eot     equ     4
0011    xon     equ     $11
0013    xoff    equ     $13
0005    ctle    equ     $05     returns to menu
001a    ctlz    equ     $1a     end of file marker
002A    asth    equ     '*
000D    cr      equ     $d
000C    ff      equ     $c      cursor home & screen clear character

0000            org     0
        * Page 0 storage used for variables *

0000    bufptr  rmb     2       pointer to character position in buffer
0002    filend  rmb     1       flag set when end of file found
0003    chksum  rmb     1       checksum
0004    errcnt  fcb     0       error counter
0005    xtemp   rmb     2       temporary 6 register stores
0007    xtemp1  rmb     2       "
0009    xtemp2  rmb     2       "
000B    xtemp3  rmb     2       "
000D    stakst  rmb     2       stack pointer store
000F    stloc   rmb     2       pointer to starting string location
0011    endloc  rmb     2       pointer to ending string location
0013    name    rmb     8       filename temporary buffer
001B    ext     rmb     3       extension temp buffer
001E    endbfl  rmb     2       pointer to end of input buffer

        * VARIOUS FLAGS *

0020    rcvbfl  rmb     1
0021    filag   rmb     1
0022    comflg  rmb     1
0023 00 loop    fcb     0
0024 00 altflg  fcb     0
0025 00 allpr   fcb     0
0026 00 abtflg  fcb     0
0027 00 pthnam  fcb     0
0028 00 qdflg   fcb     0

        * SWITCHES
        0029    swsw    equ     *
0029 00         batch   fcb     0       set for non-batch file transfers
002A 00         scrlog  fcb     0       display log off
002B 00         echo    fcb     0       local echo off
        002C    endsw   equ     *       make sure we don't get to CRCC until implemented

002C 00         crcc    fcb     0       cyclic redundancy check off

0100            org     $100

0100 20 03      start   bra     start1
0102 01         vn      fcb     1       version number
0103 2E         fcc     '.'
0104 00         fcb     0

        ***** PROGRAM STARTS HERE *****
0105 100F 00    start1  sts     stakst  save stack pointer
0108 10CE 0F33          lds     #stack  initialise stack
010C 00   39            bsr     init    initialise modem port
010E BE   0178          lds     #signon print signon message & MENU
0111 BD   CD1E          jsr     pstrng
0114 BD   CD15  getcmd  jsr     getch   get menu selection
0117 84   5F            anda    #$5F    case to upper case
0119 8E   0144          lds     #cmdtbl go find command
011C E6   84    getc1   ldab    0,x
011E E1   04            cmpb    04
0120 26   08            bne     getc2   not found but not table end
0122 8E   0787          lds     #notcmd selection not found. Print menu again
0125 BD   CD1E          jsr     pstrng
0128 20   EA            bra     getcmd
012A 34   04 A1E0 getc2  cba
012E 26   0F            bne     getc3
0130 AE   01            lds     1,x
0132 AD   84            jsr     0,x     go do command
0134 BD   CD24          jsr     pcrlf
0137 8E   0787          lds     #notcmd back to menu again
013A BD   CD1E          jsr     pstrng
013D 20   05            bra     getcmd
013F 30   01    getc3   ins
0141 30   01            ins
0143 30   01            ins
0145 20   D5            bra     getc1   next entry in command table

        ***** OFTEN USED SUBROUTINES *****

        ************************
        *   INITIALISE ACIA    *
        * initialise serial port *
        ************************
0147 B6   E003  init    ldaa    #$3
0149 B7   E00A          staa    aciac
014C B6   15            ldaa    #$15    set for 8 data bits, 1 stop, no parity
014E B7   E00A          staa    aciac
0151 39                 rts

        ***************************************
        *       MODEM OUTPUT STATUS           *
        * sets carry flag when able to        *
        * accept character. B reg destroyed   *
        ***************************************
0152 F6   E00A  modout  ldab    aciac   get modem status
0155 57                 asrb
0156 57                 asrb
0157 39                 rts

        ***************************************
        *       MODEM INPUT STATUS            *
        * sets carry flag when character      *
        * found. B reg destroyed              *
        ***************************************
0158 F6   E00A  modinp  ldab    aciac   get modem status
015B 57                 asrb
015C 39                 rts

        ***************************************
        *       KEYBOARD STATUS               *
        * sets carry flag when character      *
        * available from keyboard. B reg      *
        * destroyed                           *
        ***************************************
015D F6   E008  kbdst   ldab    kbd     get keyboard status
0160 57                 asrb
0161 39                 rts

        ***************************************
        *          MODEM OUTPUT               *
        * outputs character in A reg. All     *
        * registers preserved                 *
        ***************************************
```

```
0162 B7  E08D    modout  staa   acia4d
0165 39                  rts

                 ********************************
                 *          MODEM INPUT         *
                 * returns character from line in A *
                 * reg. All registers preserved     *
                 ********************************
0166 B6  E08D    modin   ldaa   acia4d
0169 39                  rts

                 ********************************
                 *         KEYBOARD INPUT       *
                 * returns character from keyboard in *
                 * A. All registers preserved       *
                 ********************************
016A B6  E089    kbdin   ldaa   kbd+1
016D 39                  rts

                 ********************************
                 *           TIMEOUT            *
                 * If no input from line, fflag     *
                 * cleared. 'B' reg has number of   *
                 * seconds taken to timeout on entry *
                 * fflag set if character found     *
                 *      B and X destroyed           *
                 ********************************
016E 7F  21      timout  clr    fflag
0170 8E  3B40    tmot1   ldx    #15d40    1 second at 1MHz
0173 34  04              psh    b
>0175 BD  0150           jsr    kbdst     anything from keyboard?
0178 35  04              pul    b
017A 24  0F              bcc    tmot2
017C 8D  EC              bsr    kbdin     go get it
017E 81  05              cmpa   #clld     back to main?
0180 26  09              bne    tmot2
>0182 10CE 0040          lds    #stakst   reset stack pointer
0186 BD  D403            jsr    fmcls     close any files left open
0189 70  15              dec    tmot5
018B 34  04      tmot2   psh    b
>018D BD  015B           jsr    endist    anodes character received?
0190 35  04              pul    b
0192 25  09              bcs    tmot3     yes - continue
0194 30  1F              des               decrement til 1 second done
0196 26  F3              bne    tmot2
0198 5A                  decb             repeat required number of times
0199 26  05              bne    tmot1
019B 20  02              bra    tmot4     back to calling routine
019D 03  21      tmot3   com    fflag     set character found flag
019F 39          tmot4   rts
01A0 7E  0114    tmot5   jmp    gotcmd    back to main menu

                 ********************************
                 *    TRANSMIT CHARACTER IN A       *
                 * checks output port status. Loops *
                 * till free. B register destroyed  *
                 ********************************
01A3 BD  A0      tschr   bsr    aodout    check acia status
01A5 24  FC              bcc    tschr     loop til free
01A7 BD  89              bsr    amdout    transmit character
01A9 39                  rts

                 ********************************
                 *     RECEIVE CHARACTER IN A       *
                 * checks input port status and     *
                 * loops till character available   *
                 * gets character from line,        *
                 * displays it & returns with it in *
                 * A. B register destroyed          *
                 ********************************
01AA BD  AC      rschr   bsr    aodist    test acia status
01AC 24  FC              bcc    rschr     loop til free
01AE BD  B6              bsr    aodin     get character in A
01B0 34  02              psh    a
0182 BD  CD1B            jsr    putchr    display it
01B5 35  02              pul    a
01B7 39                  rts

                 ********************************
                 *       COMPUTE CHECKSUM       *
                 * adds A to chsum. All regs presvd *
                 ********************************
01B8 34  02      check   psh    a
01BA 9B  03              adda   cksum
01BC 97  03              staa   cksum
01BE 35  02              pul    a
01C0 39                  rts

                 ********************************
                 *      WAIT TIL SENDER STOPS       *
                 * looks at line for 1 second. If   *
                 * nothing received, sends NAK or C *
                 * depending on crcc flag. A & B    *
                 * destroyed.                       *
                 ********************************
01C1 C6  01      purge   ldab   #1        set 1 second timout
>01C3 BD  016E           jsr    timout
01C6 0D  21              tst    fflag     anything from line?
01C8 27  05              beq    purge1    no - can send out now
>01CA BD  0166           jsr    modin     go get character and throw away
01CD 20  F2              bra    purge     round again
01CF 0D  2C      purge1  tst    crcc      cyclic redundancy check? g?
01D1 27  04              beq    purge2    no - xpreal mode
01D3 86  43              ldaa   #'C       used instead of NAK for crc
01D5 20  02              bra    purge3
01D7 86  15      purge2  ldaa   #nak
>01D9 BD  01A3   purge3  jsr    tschr
01DC 39                  rts

                 **** MESSAGE HANDLING ROUTINES ****

                 * SETS UP TIMEOUT MESSAGE *
01DD 86  04      prnt2   ldaa   #10       calculate timout no.
01DF 34  04 A0E0         sba
01E3 8A  30              oraa   #$30      change to ascii
01E5 B7  0B1C            staa   msg2a     put in message
01E8 8E  0013            ldx    #msg2
01EB 34  04              psh    b
01ED BD  CD1E            jsr    pstrng
01F0 35  04              pul    b
01F2 39                  rts

                 * SETS UP DUPLICATE SECTOR MESSAGE *
01F3 B6  00C0    prnt3   ldaa   blknum    get block number

01F6 8A  30              oraa   #$30      make it ascii
01F8 B7  0039            staa   msg3a
01FB 8E  0B1E            ldx    #msg3
01FE BD  CD1E            jsr    pstrng
0201 39                  rts

                 ********************************
                 *       MOVE A BLOCK OF DATA       *
                 * B reg contains number of bytes   *
                 * stloc contains address to move from *
                 * endloc contains address to move to *
                 ********************************
0202 9E  0F      mvstrg  ldx    stloc     get starting point
0204 A6  00              ldaa   0,x       get data byte
0206 30  01              ins
0208 9F  0F              sts    stloc     update start
020A 9E  11              ldx    endloc    get end location
020C A7  00              staa   0,x       store data
020E 30  01              ins
0210 9F  11              sts    endloc    update end
0212 5A                  decb             adjust byte count
0213 26  E0              bne    mvstrg    loop til done
0215 39                  rts

                 ***** DISK FILE RELATED SUBROUTINES *****

                 ********************************
                 *       WRITE A BLOCK TO DISK      *
                 * writes input buffer to disk file *
                 * in standard FLEX format. NO: all *
                 * ascii chars allowed, but no      *
                 * binary file info stored          *
                 ********************************
0216 8E  00C0    wfil    ldx    #bufst    point to start of data
0219 9F  00      wfil1   stx    bufptr
021B A6  84              ldaa   0,x       get char from buffer
021D 8E  0C4F            ldx    #fcb      point to FCB
0220 BD  D406            jsr    fmswr     write to file
0223 26  0A              bne    wfil2     check errors
0225 9E  00              ldx    bufptr
0227 30  01              ins              update pointer
0229 8C  0C4D            cpx    #bufend+1 end of data?
022C 26  EB              bne    wfil1     continue til out of data
022E 39                  rts
>022F 7E  0230   wfil2   jmp    error     report errors

                 ********************************
                 *          CLOSE FILE          *
                 * close file whose name is in fcb & *
                 * print file closed message        *
                 ********************************
0232 86  04      fcls    ldaa   #4        file close
0234 8E  0C4F            ldx    #fcb
0237 A7  00              staa   0,x
0239 BD  D406            jsr    fmswr     go close file
023C 86  0290            ldaa   #vmlnam   get filename into printstring
023F C6  07              ldab   #7
0241 8E  0B8C            ldx    #msg9+1   'closed'
0244 9F  0F              stx    stloc
0246 8E  0BA0            ldx    #msg8f
0249 9F  11              stx    endloc
>024B BD  0202           jsr    mvstrg    set up rest of string
024E 8E  0B9D            ldx    #msg8a    let people know file closed
0251 BD  CD1E            jsr    pstrng
0254 39                  rts

                 ********************************
                 *          OPEN FILE           *
                 * A register contains file open    *
                 * code on entry - 1 for read, 2 for *
                 * write. Prompts for filename (FLEX *
                 * format - defaults to .TXT)       *
                 ********************************
0255 34  02      fopen   psh    a
0257 8E  0AA9            ldx    #msg       prompt for file name
025A BD  CD1E            jsr    pstrng
025D BD  CD1B            jsr    inbuf      get filename into line buffer
0260 8E  0C4F            ldx    #fcb
0263 BD  CD28            jsr    getfil
0266 86  01              ldaa   #1         default extension is TXT
0268 BD  CD33            jsr    valext
026B 35  02              pul    a
026D A7  84              staa   0,x
026F BD  D406            jsr    fmswr     open file
0272 27  02              beq    fopen1    report any errors
0274 20  04              bra    fopen2
0276 BD  CD2A    fopen1  jsr    pcrlf
0279 39                  rts
>027A 7E  0270   fopen2  jmp    error     error

                 ********************************
                 *      REPORT FILE ERRORS      *
                 * reports file errors in FLEX style *
                 * and returns to mem               *
                 ********************************
027D BD  CD3F    error   jsr    rpterr
0280 10CE 0F33           lds    #stack    reset stack pointer
0284 BD  CD2A            jsr    pcrlf
0287 9E  09D7            ldx    $09d7     print name
028A BD  CD1E            jsr    pstrng
028D 7E  0114            jmp    gotcmd

                 ********************************
                 *       MOVE FILENAME          *
                 * moves filename from fcb to print *
                 * string                           *
                 ********************************
0290 C6  08      mvlnam  ldab   #8         8 characters to move
0292 8E  0C53            ldx    #fcb+4    file name in fcb
0295 9F  0F              stx    stloc
0297 8E  0B9D            ldx    #msg8a    file message string
029A 9F  11              stx    endloc
029C BD  0202            jsr    mvstrg
029F C6  03              ldab   #3         3 chars in xtnsn
02A1 8E  0C58            ldx    #fcb+12   point to extent in fcb
02A4 9F  0F              stx    stloc
02A6 8E  0BA6            ldx    #msg8c
02A9 9F  11              stx    endloc
02AB BD  0202            jsr    mvstrg
02AE 39                  rts

                 ***** RETURN TO FLEX *****
```

```
02A6 1F0E 02      qofirs  lds     stabat    reset block pointer
02A9 8D  D483             jsr     fcscls    close any files left open
02AC 7E  C003             jmp     warms     files were start entry

                  ***** FILE RECEIVE SECTION *****

                  * READ BLOCK INTO BUFFER *
02AF 0E  00CA     rblok   ldi     #sob      start of buffer
02B1 9F  05               sts     xtemp     save buffer pointer
02B3 8D  26               bsr     rech      go get character from line
02B5 81  18               cmpa    #can      cancel?
02B7 26  04               bne     rblok1    no - continue
02B9 03  72               com     canflg    yes - set flag & return
02BB 20  1B               bra     rblok3    end of file?
02BD 81  04       rblok1  cmpa    #eot
02BF 26  04               bne     rblok2    no - continue
02C1 0F  F2               clr     filend    yes - set flag & return
02C3 20  13               bra     rblok3
02C5 9E  05       rblok2  ldx     xtemp     restore buffer pointer
02C7 A7  00               staa    0,x       store in buffer
02C9 9C  1E               cpx     #eoblk    end of block?
02CB 27  08               beq     rblok3    yes - return
02CD 8D  0180             jsr     check     no - calculate checksum
02D0 30  01               ins
02D2 9F  05               sts     xtemp     save buffer pointer
02D4 08  E3               bne     rech      continue filling buffer
02D6 20  ED               bra     rblok2
02D8 39           rblok3  rts

                  * READ CHARACTER WITHOUT ECHO TO SCREEN *
02D9 C6  0A       rech    ldab    #10       set 10 second timeout
02DB 8D  016E             jsr     tinout    get input through timout routine
02DE 0D  21               tst     fflag     anything there?
02E0 27  05               beq     rech1     no - error condition
02E2 8D  0166             jsr     acdin     go get char
02E5 20  03               bra     rech2
02E7 8E  0E0D     rech1   lds     #stack    will force error condition
02EA 39           rech2   rts

02EB 39           crchk   rts               *** crcc check routine here ***

                  **** RECEIVE IN BATCH MODE ****

                  * PREPARE TO RECEIVE FILE-NAME *
02EC 9F  05       setxt   sts     xtemp     save index reg
02EE DE  0C52             ldx     #fcb+1    point to start of filename in fcb
02F0 B6  CE0C             ldaa    workdr    get new file on working drive
02F3 A7  04               staa    4,x
02F5 30  01               ins
02F7 9F  07               sts     xtemp1    save start of actual filename
02F9 86  0B               ldaa    #null
02FB A7  04       setxt1  staa    0,x       clear out old filename
02FD 30  01               ins
02FF 8C  0C5E             cpx     #fcb+45   end of filename?
0302 26  F7               bne     setxt1
0304 86  01               ldaa    #1        make default extension 'TXT'
0306 8E  0C4F             lds     #fcb
0309 8D  C053             jsr     setext
030B 39                   rts

                  * MAKE FIRST CHAR OF FILENAME A EXT ALPHA *
030C 9E  05       first   ldx     xtemp     point to buffer
030E A6  04       first0  ldaa    0,x
0310 81  41               cmpa    #'A       check for alpha
0312 2D  0C               blt     first2
0314 81  5A               cmpa    #'Z
0316 2F  0A               ble     first3
0318 81  61               cmpa    #'a
031A 2D  04               blt     first2
031C 81  7A               cmpa    #'z
031E 2F  02               ble     first3
0320 86  7A       first2  ldaa    #'z       change any others to 'z'
0322 39           first3  rts

                  * MAKE REMAINDER OF NAME FILE COMPATIBLE *
0323 81  20       next    cmpa    #' 
0325 27  14               beq     next2
0327 2D  10               blt     next1
0329 81  5B               cmpa    #'[
032B 2D  0C               blt     next1
032D 81  5F               cmpa    #'_       underscore allowed
032F 22  06               bhi     next2
0331 8D  07               bsr     first1    do alpha comparison
0333 20  02               bra     next2     changed to 'A' if not alpha
0335 86  7A       next1   ldaa    #'z       takes care of early failures
0337 39           next2   rts

                  * FILE RECEIVE ROUTINE *
0338 86  02       frecv   ldaa    #2        set for write
033A BD  0255             jsr     fopen     open file
033D 8D  0166             jsr     acdin     dummy read to clear acia
0340 0F  02               clr     fsiend    end of file flag
0342 0F  22               clr     canflg    cancel transfer flag
0344 8D  01C1     frecv0  jsr     purge     send initial nak
0347 86  01               ldaa    #1        look for 1st block
0349 97  28       frecv1  staa    rcvblk
034B 0F  03               clr     ch3ma
034D C6  0A       frcv15  ldab    #10       set retry limit at 10
034F 34  04       frecv2  psh     b
0351 C6  0A               ldab    #10       set 10 second timeout
0353 8D  016E             jsr     tinout    get char through timout routine
0356 0F  20               clr     gdflg     assume bad file transfer
0358 35  04               pul     b
035A 0D  21               tst     fflag     char available?
035C 26  08               bne     frecv3    go get it
035E 0B  01D0             jsr     prnt2     timeout message
0361 5A                   decb
0362 26  EB               bne     frecv2    continue til no more timeouts
0364 20  71               bra     frecv7
0366 8D  02D0     frecv3  jsr     rblok     go get block
0369 0D  22               tst     canflg    did sender cancel?
036B 27  08               beq     frcv32
036D 8E  0037             lds     #msg4
0370 8D  E01E             jsr     putmsg
0373 20  62               bra     frecv7
0375 0B  02       frcv32  tst     fileod    end of file?
0377 27  09               beq     frcv35
0379 86  06               ldaa    #ack      send ack & close up
037B 8D  01A3             jsr     tocho
037E 43  20               com     gdflg     good transfer
0380 20  55               bra     frecv7

0382 96  28       frcv35  ldaa    rcvblk    get our block number
0384 01  00C0             cmpa    blkmsw    same as that received?
0387 27  28               beq     frecv4
0389 96  29               ldaa    rcvblk    no - might be repeat of previous block
038B 4A                   deca
038C 81  00C8             cmpa    blkmsw    same?
038F 26  17               bne     frcv37    no - must be dud
0391 0E  00CA             lds     #sob      make sure no corruption
0394 A6  04               ldaa    0,x
0396 AB  01               adda    1,x
0398 AB  02               adda    2,x
039A 26  0C               bne     frcv37    good number would give zero
039C D0  0103             jsr     prnt3     duplicate sector message
039F 86  06               ldaa    #ack      acknowledge receipt
03A1 8D  01A3             jsr     tocho
03A4 96  28               ldaa    rcvblk    then throw block away
03A6 20  A1               bra     frecv1
03A8 8D  01C1     frcv37  jsr     purge     we want this block again
03AB 20  A2               bra     frecv2
03AD 0D  2C       frecv4  tst     crcc      cyclic redundancy flag set?
03B0 27  03               beq     frecv5    no - do checksum
03B2 8D  02F5             jsr     crchk
03B5 F6  00C0     frecv5  ldab    chblk     compare checksums
03B8 C1  03               cmpb    cksum
03BA 27  05               beq     frecv6
03BC 8D  01C1             jsr     purge     send block again
03BF 20  98               bra     frcv15
03C1 8D  0210     frecv6  jsr     wfil      go write buffer full to disk
03C4 86  3A               ldaa    #astk
03C6 8D  C010             jsr     putchr
03C9 86  06               ldaa    #ack      then acknowledge
03CB 8D  01A3             jsr     tocho
03CE 0D  02               tst     fsiend    look for end of file
03D0 26  06               bne     frecv7    yes - close up & return
03D2 96  28               ldaa    rcvblk
03D4 4C                   inca              increment block count
03D5 7E  0359             jmp     frecv1
03D8 8D  0232     frecv7  jsr     fcls      close file
03DB 39                   rts

                  * BATCH RECEIVE ROUTINE *
03DC 8D  0166     rbatch  jsr     acdin     dummy read
03DF 0F  02               clr     fsiend    clear flags
03E1 0F  22               clr     canflg
03E3 8D  C024             jsr     perif
03E6 0F  28       rbat1   clr     rcvblk    set block number to zero
03E8 0F  03               clr     cksum
03EA C6  0A               ldab    #10       set retry limit to 10
03EC 34  04       rbat2   psh     b
03EE 8D  01C1             jsr     purge     send initial nak
03F1 C6  0A               ldab    #10       10 second timeout
03F3 8D  016E             jsr     tinout    wait for input
03F6 35  04               pul     b         restore retry number
03F8 0D  21               tst     fflag
03FA 26  09               bne     rbat3     continue if char found
03FC 8D  0100             jsr     prnt2     timeout message
03FF 5A                   decb
0400 26  EA               bne     rbat2     try 10 times
0402 7E  04A0             jmp     rbat10
0405 8D  02B0     rbat3   jsr     rblok
0408 0D  22               tst     canflg
040A 27  03               beq     rbat4     continue unless sender cancels
040C 7E  04A0             jmp     rbat10
040F 96  28       rbat4   ldaa    rcvblk
0411 81  00C0             cmpa    blkmsw
0413 27  02               beq     rbat6     check for block zero
0415 20  CE       rbat5   bra     rbat1
0417 F6  00C0     rbat6   ldab    chblk
041A C1  03               cmpb    cksum
041C 27  02               beq     rbat7     continue if no errors in block
041E 20  C5               bra     rbat1
0420 86  06       rbat7   ldaa    #ack      acknowledge good block
0422 8D  01A3             jsr     tocho
0425 BE  00C0             lds     #bufst    point to 1st received char
0428 A6  04               ldaa    0,x       get it
042A 81  00               cmpa    #null     is at a null?
042C 27  06               beq     rbat10    end of transfer
042E 8D  02F6             jsr     setxt     set extension etc
0431 8D  0318             jsr     first     make first letter 'z' if not alpha
0434 0E  0C53             lds     #fcb+4
0437 A7  04               staa    4,x
0439 30  01               ins
043B 9F  07               sts     xtemp1
043D 9E  05               lds     xtemp
043F 30  01               ins
0441 A6  04       rbat8   ldaa    0,x
0443 81  00               cmpa    #null
0445 27  21               beq     rbat9     continue if end of filename
0447 81  2E               cmpa    #'.       look for start of extension
0449 27  11               beq     rbat8a
044B 8D  032F             jsr     next      make filename files compatible
044E 9F  05               sts     xtemp
0450 9E  07               lds     xtemp1
0452 A7  04               staa    4,x       store in fcb
0454 30  01               ins
0456 9F  07               sts     xtemp1
0458 9E  05               lds     xtemp
045A E3  03               addd    #3        square '.'
045C 9F  05               sts     xtemp
045E DE  0C50             lds     #fcb+12
0461 9F  07               sts     xtemp1
0463 8D  0510     rbat8a  jsr     first
0466 20  D7               bra     rbat8     continue parsing extension
0468 DE  0C4F     rbat9   lds     #fcb
046B 86  02               ldaa    #2        open file for writing
046D A7  00               staa    0,x
046F BD  3406             jsr     fos
0472 27  02               beq     rbat9a
0474 20  1F               bra     rbat11
0476 8D  0298     rbat9a  jsr     avfnam    get filename
0479 C6  07               ldab    #7
047B 8E  00C3             lds     #msg9+14  'created'
047E 9F  0F               sts     stloc
0480 8E  00AD             lds     #msg8f
0483 9F  11               sts     endloc
0485 8D  02D2             jsr     avstrg    finish string set up
0488 8E  0090             lds     #msg8a    print it
048B BD  E01E             jsr     putmsg
048E BD  0354             jsr     frecv0    go get file
0491 7E  03E0             jmp     rbatch    continue til null pathname recvd
0494 39           rbat10  rts
0495 7E  027D     rbat11  jmp     error     report errors

                  **** SET RECEIVE MODE ****
```

```
0440 DE  0C40    filrec  ldx   #ckblk    end of normal block
0442 00  2C              tst   crcc
0444 27  02              beq   filr1
0446 30  01              ins             crcc uses ultra character
0448 9F  1E      filr1   stx   endblk
044A 0D  29              tst   batch
044C 26  05              bne   filr2
044E BD  034B            jsr   frecv
0451 20  03              bra   filr3
0453 BD  03E9    filr2   jsr   rbatch
0456 4D  28      filr3   tst   qdflg
0458 27  0A              beq   filr4     bad file receipt
045A DE  097F            ldx   #msg7
045C BD  CD1E            jsr   putmsg
045F 39          filr4   rts

                 ***** TERMINAL MODE *****

                 * BUILD BUFFER FROM SCREEN I/O *
0460 9E  00      build   ldx   bufptr    x hi to next free space
0462 A7  84              staa  0,x       store character
0464 8C  0C47            cpx   #bufend-5 almost end of buffer?
0467 26  34              bne   build3
0469 86  13              ldaa  #xoff     halt input from line
046B BD  01A3            jsr   txchr
046E BD  016E    build0  jsr   timout    get any new char through timeout
0471 7E  00              ldx   bufptr
0473 8C  0C4C            cpx   #bufend
0476 27  1A              beq   build4
0478 0D  21              tst   fflag     char found?
047A 27  09              beq   build2
047C BD  01AE            jsr   rxchr1    go get it
047F 30  01              ins
0481 A7  84              staa  0,x
0483 9F  00              stx   bufptr    save pointer
0485 2F  E7              bra   build1    yes if any more coming
0487 86  00      build2  ldaa  #null     fill remainder of buffer with nulls
0489 30  01              ins
048B A7  84              staa  0,x
048D 8C  0C4C            cpx   #bufend
0490 26  F5              bne   build2
0492 BD  021A    build4  jsr   wfil      write buffer to disk
0495 DE  0DCC            ldx   #bufst-1  reset pointer to start of buffer
0498 86  11              ldaa  #xon     restart transmission from other end
049A BD  01A3            jsr   txchr
049D 30  01      build3  ins             update pointer
049F 9F  00              stx   bufptr
04A1 39                  rts

                 * TALK MODE - DON'T LOG DATA *
04A2 BD  CD24    talk    jsr   pcrlf
04A5 BD  015D    talk05  jsr   kbdst     test keyboard status
04A8 24  11              bcc   talk1     nothing pending
04AA BD  d16A            jsr   kbdin     get character from kbd
04AD 81  05              cmpa  #ctle     is it control E?
04AF 27  1B              beq   talk3     back to comm
04B1 BD  01A3            jsr   txchr     send it to line
04B4 7D  0020            tst   echo
04B7 27  03              beq   talk1
04B9 BD  CD1B            jsr   putchr    display character
04BC BD  015D    talk1   jsr   kbdst     anything coming in?
04BF 26  E5              bcc   talk05    loop til some activity
04C1 BD  01AE            jsr   rxchr1    get input & display it
04C4 7D  0020            tst   echo
04C7 27  07              beq   talk2
04C9 BD  01A3            jsr   txchr     send it back out
04CC 20  DF              bra   talk05    around again
04CE 39          talk3   rts

                 * LOG SCREEN OUTPUT TO DISK *
04CF BD  CD24    log     jsr   pcrlf
04D2 86  02              ldaa  #2       set for write
04D4 BD  0255            jsr   logon     go open file
04D7 DE  0DCD            ldx   #bufst    point to data store
04D9 9F  00              stx   bufptr
04DB BD  015D    log1    jsr   kbdst     test for local character
04DE 24  10              bcc   log2      no - look for input from line
04E0 BD  d16A            jsr   kbdin     get char from keyboard
04E3 81  05              cmpa  #ctle     is it control E
04E5 27  1E              beq   log3      close file & return
04E7 BD  01A3            jsr   txchr     send it to line
04EA 7D  0020            tst   echo      echo?
04ED 27  04              beq   log2
04EF 34  02              psh   a         save for later
04F1 BD  CD1B            jsr   putchr    display it
04F4 35  02              pul   a         restore it
04F6 BD  04CA            jsr   build     store in buffer
04F9 09  015D    log2    jsr   kbdst     look for input from line
04FC 24  DE              bcc   log1      no - loop around again
04FE BD  01AE            jsr   rxchr1    go get it
0501 BD  04CA            jsr   build     store in buffer
0504 20  D6              bra   log1
0506 9E  00      log3    ldx   bufptr    get pointer
0508 30  1F              des             adjust it
050A 9F  00              stx   bufptr    resave pointer
050C BD  04FE            jsr   build2    fill remainder of buffer with nulls
050F BD  0232            jsr   fcls
0512 39                  rts

                 * SET LOG MODE IF REQUIRED *
0513 0D  2A      term    tst   scrlog
0515 26  01              bne   term1
0517 7E  050E            jmp   talk
051A 7E  0537    term1   jmp   log

                 ***** FILE TRANSMIT SECTION *****

                 * FILL TRANSMIT BUFFER FROM DISK FILE *
051D DE  0DCD    fbuff   ldx   #bufst    point to start of buffer
051F 9F  00              stx   bufptr
0521 DE  0C4F    fbuff1  ldx   #fcb      get character from file
0523 BD  1486            jsr   fgs
0526 26  0F              bne   fbuff2    test error status
0528 9E  00              ldx   bufptr
052A A7  84              staa  0,x       store char in buffer
052C 30  01              ins             increment pointer
052E 9F  00              stx   bufptr    and store it
0530 8C  0C4D            cpx   #bufend+1 128 bytes?
0533 26  E8              bne   fbuff1    no - get next char
0535 20  1E              bra   fbuff4    exit - buffer full
0537 A6  01      fbuff2  ldaa  1,x       get error number
0539 81  08              cmpa  #8        is it end of file?
053B 27  02              beq   fbuf25    no - report error
053D 20  17              bra   fbuf15
```

```
053F 86  06      fbuf25  ldaa  #6        close file
0541 A7  04              staa  0,x
0543 BD  CD06            jsr   fos
0546 8A  1A              ldaa  #sub     fill remainder of buffer
0548 9E  00              ldx   bufptr    with control Z
054A A7  04      fbuf3   staa  0,x
054C 30  01              ins
054E 8C  0C4D            cpx   #bufend+1
0551 26  F7              bne   fbuf3
0553 03  02              com   filend    set end of file flag
0555 39          fbuf4   rts
0556 7E  0270    fbuf15  jmp   error

                 * TRANSMIT A BLOCK *
0559 DE  0BCA    sblok   ldx   #sob     block beginning
055C A6  84      sblok1  ldaa  0,x
055E BD  0189            jsr   chout     compute checksum
0561 BD  01A3            jsr   txchr     send character
0564 8C  0C4C            cpx   #bufend   last in buffer?
0567 27  04              beq   sblok2    yes - go again
0569 30  01              ins
056B 20  EF              bra   sblok1
056D 9E  2C      sblok2  lst   crcc      cyclic redundancy check?
056F 27  03              beq   sblok3    no - send checksum
0571 BD  05EC            jsr   crcalc    calculate crcc and send it
0574 96  03      sblok3  ldaa  chksum   get block check character
0576 BD  01A3            jsr   txchr     send it
0579 BD  0166            jsr   modin     clear receiver
057C 86  2A              ldaa  #astk     print asterisk for each block received
057E BD  0010            jsr   putchr
0581 39                  rts

0582 39          crcalc  rts             *** cyclic redundancy send routine here ***

                 * TRANSMIT FILE *
0583 86  01      txfil   ldaa  #1        first block
0585 20          clr   qdflg     assume bad transfer
0587 BF  03      txfil1  clr   chksum    new block starts with clear checksum
0589 4F  04              clr   ercnt     no errors at this stage
058B 97  0BCB            staa  blknum    block number
058D 43                  coma            block number check
058E 97  0BCC            staa  blkchk
0590 BD  05  C5          bsr   sblok     send this block
0593 C6  0A              ldab  #10       10 second timeout
0595 BD  016E            jsr   timout    get ACK through timout
0598 D0  21              tst   fflag
059A 27  07              beq   txfil25
059C BD  0166            jsr   modin
059F 81  06              cmpa  #ack
05A1 27  16              beq   txfil4    continue - block received OK
05A3 96  04      txfil25 ldaa  ercnt
05A5 4C                  inca
05A6 97  04              staa  ercnt
05A8 81  05              cmpa  #5        only 5 tries allowed
05AA 26  E5              bne   txfil2    abort filesend after 5 tries
05AC 0E  00A5            ldx   #msg4     "abort...."
05AF BD  CD1E            jsr   putmsg
05B2 BD  0232            jsr   fcls      close files and return
05B5 20  39              bra   txfil7
05B7 20  DB      txfil3  bra   txfil2    retransmit block
05B9 0F  04      txfil4  clr   ercnt     reset error counter
05BB 00  02              tst   filend    end of file?
05BD 26  0F              bne   txfil5     go do wind up
05BF BD  05B5            jsr   fbuff     get next block from disk
05C2 86  0BCB            ldaa  blknum    increment block number
05C5 4C                  inca
05C6 20  0E              bra   txfil1     transmit new block
05C8 09  27      txfil5  tst   pthnam    was it a header block?
05CA 26  24              bne   txfil7     don't send EOT
05CC 86  04              ldaa  #eot      ASCII EOT ends file transfer
05CE BD  01d1            jsr   txchr
05D1 C6  0A              ldab  #10       10 second timeout
05D3 BD  016E            jsr   timout    anything there?
05D6 27  0F              beq   txfil6    no - try again?
05D8 BD  0166            jsr   modin
05DB 81  06              cmpa  #ack      good char?
05DD 26  04              bne   txfl55    yes - all done folks
05DF 03  29              com   qdflg     good transfer
05E1 20  11              bra   txfil7
05E3 81  15      txfl55  cmpa  #nak
05E5 27  0F              beq   txfil5    try again
05E7 96  04      txfil6  ldaa  ercnt
05E9 4C                  inca
05EA 81  05              cmpa  #5        only 5 tries allowed
05EC 26  B0              bne   txFil5
05EE 39          txfil7  rts

                 * MOVE FILENAME TO FCB BUFFER AND PRINT STRING *
05EF C6  0C      movnam  ldab  #12       move filename from directory
05F1 DE  0092            ldx   #dirfcb+3 drive number to directory
05F4 9F  0F              stx   stloc
05F6 DE  0C32            ldx   #fcb+3    drive number in fcb
05F9 9F  11              stx   endloc
05FB BD  0202            jsr   mvstrg    move drive number & filename
05FE C6  08              ldab  #8        move filename into print string
0600 DE  00e3            ldx   #dirfcb+6 filename start
0603 9F  0F              stx   stloc
0605 DE  009B            ldx   #msgBk
0608 9F  11              stx   endloc
060A BD  0202            jsr   mvstrg
060D C6  03              ldab  #3        move extension into print string
060F DE  0099            ldx   #dirfcb+12 extension start
0612 9F  0F              stx   stloc
0614 DE  0BAb            ldx   #msgBk
0617 9F  11              stx   endloc
0619 BD  0202            jsr   mvstrg
061C C6  09              ldab  #13       put filename in FAR format into buffer
061E DE  0099            ldx   #msgBk
0621 9F  0F              stx   stloc
0623 DE  0DCD            ldx   #bufst
0626 9F  11              stx   endloc
0628 BD  0202            jsr   mvstrg
062B C6  0C              ldab  #12       replace nulls with spaces (null ends filename in FAR)
062D DE  0DCD            ldx   #bufst
0630 A6  84      movnul  ldaa  0,x
0632 81  00              cmpa  #0        CP/M has spaces not nulls
0634 26  04              bne   movnul2
0636 86  20              ldaa  #$20      change nulls to spaces
0638 A7  04              staa  0,x
063A 30  01      movnul2 ins
063C 5A                  dcb
063D 26  F1              bne   movnul    continue til all changed
063F 39                  rts
```

```
                        * LOAD ERROR CODE INTO FCB & REPORT IT *
06A0 8E   0D0F  serr    ldx   #dirfcb
06A0 C6   1A            ldab  #26        status error
06A2 E7   01            stab  1,x
06A4 7E   027B          jmp   error       report error & return to menu

                        * CHECK FOR FLEX LINE TERMINATION CHARACTER *
06A7 96   CC10  tsttrm  ldaa  lstchr      check last terminator
06A9 81   0D            cmpa  #$d        carriage return?
06AB 27   03            beq   tsttr2      yes - return
06AD 81   CC02          cmpa  eol         FLEX end of line character?
06AF 39         tsttr2  rts

                        * END BATCH TRANSFER *
06B0 7F   0DCB  exit    clr   burst      set null file name
06B3 03   26            com   abtflg     flag transfer complete
06B5 BD   B70E          jsr   send       send last block
06B8 39                 rts


                        * GET DIRECTORY ENTRY *
06B9 8E   0D0F  getdir  ldx   #dirfcb    point to directory fcb
06BC 86   07            ldaa  #7         get information record
06BE A7   04            staa  0,x
06C0 7E   D40A          jmp   fas

                        * GET FILENAME FROM LINE BUFFER *
06C3 0F   24    getnam  clr   allflg     clear all
06C5 8E   0D13          ldx   #name      filename buffer
06C8 C6   0B            ldab  #11        clear it
06CA 6F   84    getna2  clr   0,x
06CC 30   01            ins
06CE 5A                 decb
06CF 26   F9            bne   getna2
06D1 BE   CC14          ldx   bufpnt     line buffer pointer
06D4 A6   04            ldaa  0,x
06D6 81   2E            cmpa  #'.        just extent?
06D8 27   05            beq   getna4     yes - go get it
06DA BE   0D0F          ldx   #dirfcb    dummy fcb
06DD 20   1A            bra   getna7
06DF 7C   CC15  getna4  inc   bufpnt+1   point to 1st char of ext
06E2 C6   03            ldab  #3         go get it
06E4 8E   0D1B          ldx   #ext
06E7 BD   CD27  getna5  jsr   nxtch
06EA 25   0A            bcs   getna6     check for any more in ext
06EC A7   04            staa  0,x
06EE 20   01            bra
06F0 5A                 decb
06F1 2A   F4            bpl   getna5
06F3 1A   01            sec
06F5 39                 rts
06F6 0C   FE    getnam  clc
06F8 39                 rts
06F9 7E   CD2D  getn07  jmp   getfil     get name from line buffer

                        * SEND FILE AFTER INITIAL NAK *
06FC 96   FF    send    ldaa  #$ff
06FE 97   02            staa  filend     set filend so no EOF after name
0700 97   27            staa  pthnam     set pathname flag
0702 BD   0166          jsr   nodin      dummy read to clear acia
0705 C6   05            ldab  #5         set retry limit
0707 D7   04            stab  errcnt
0709 C6   3C            ldab  #60        60 second initial timeout
070B BD   016E  send1   jsr   timout     look for input from receiver
070E 69   21            tst   rflag      anything there
0710 26   08            bne   send3      go get it
0712 64   04            dec   errcnt     adjust error count
0714 27   25            beq   send5      reset timeout to 10 seconds
0716 C6   0A            ldab  #10
0718 20   F1            bra   send1
071A BD   0166  send3   jsr   nodin      go get character
                        * for cyclic redundancy checking, receiver will *
                        * send C instead of NAK. Should be tested here  *
                        * and crcc set when crc routines available      *
071D 81   15            cmpa  #nak       initial NAK?
071F 26   F1            bne   send2      no - try again
0721 4F                 clra             set block number to zero
0722 BD   05F1          jsr   fillll     go send pathname block
0725 69   26            tst   abtflg     did receiver abort?
0727 26   12            bne   send5
0729 0F   02            clr   filend     set up normal conditions
072B 0F   27            clr   pthnam
072D BD   0505  send4   jsr   fbuff     read file into buffer
0730 BE   0AFC          ldx   #ostq1     initial ack
0733 BD   CD1E          jsr   putreq
0736 C6   0A            ldab  #10        10 second timeout first
0738 BD   07EE          jsr   tsend2     send file normally
073B 39         send5   rts


                        ********************************
                        *  MATCH DIRECTORY & INPUT NAMES *
                        * NOTE: routines based on TSC pdel *
                        * FLEX utility                     *
                        ********************************
073C 8E   0D0F  sfil    ldx   #dirfcb    directory fcb
073F C6   06            ldaa  #6         open directory
0741 A7   04            staa  0,x
0743 BD   D40A          jsr   fas
0746 27   02            beq   sfil1      test for errors
0748 20   70            bra   sfill1
074A 0C   25    sfil1   inc   loop
074C BD   06CD  sfil2   jsr   getdir     get information record
074F 27   03            beq   sfil3      test for errors
0751 7E   07AD          jmp   sfil0
0754 8E   0D0F  sfil3   ldx   #dirfcb
0757 A6   04            lda   0,x        look for filename
0759 26   03            bne   sfile
075B 7E   0783          jmp   sfill0     return
075E 20   EC    sfile   bra   sfil2      get next record if deleted
0760 8E   0093          ldx   #dirfcb+3
0763 BD   24            jsr   allflg     all files?
0765 26   23            bne   sfil7
0767 9F   09            stx   xtemp2     save filename pointer
0769 8E   0013          ldx   #name      test for filename match
076C 9F   00            stx   xtemp3
076E 9E   00    sfil5   ldx   xtemp3
0770 9C   001E          cpx   #name+30   end of filename buffer?
0773 27   15            beq   sfil7
0775 A4   04            ldaa  0,x
0777 3D   01            xgx
0778 9F   00            stx   xtemp1
077A 9E   09            ldx   xtemp2
077C 4B                 tstb
```

```
                        beq   sfil6      null matches anything
077D A1   04            cmpa  0,x        does it match?
077F 26   C8            bne   sfil7      no - get next record
0781 3D   01    sfil6   inx
0783 9F   09            stx   xtemp2
0785 20   E4            bra   sfil5
0787 8D   065C  sfil7   jsr   newman     get filename into fcb, buffer & printline
0789 8E   0C4F          ldx   #fcb
078C A6   01            ldaa  #1         open for read
078E A7   04            staa  0,x
0790 8D   D40A          jsr   fas
0792 27   0A            beq   sfil9      test for errors
0794 20   27            bra   sfillll
0796 A6   01    sfil0   ldaa  1,x        get error type
0798 81   00            cmpa  #0         end of directory?
079A 27   20            beq   sfill0     yes - return
079C 20   1F            bra   sfillll
079E C6   07    sfile   ldab  #7
07A0 8E   0B65          ldx   #msg7      "opened"
07A3 9F   0F            stx   stloc
07A5 8E   0BAB          ldx   #msg8f
07A8 9F   11            stx   endloc
07AA BD   0282          jsr   prstrg
07AD 8E   0B9B          ldx   #msg8a     show us what's happening
07B0 BD   CD1E          jsr   putreq
07B3 BD   B70E          jsr   send       send this file
07B6 BD   0232          jsr   fcls       now close it
07B9 7E   075E          jmp   sfil2      any more files match this name?
07BC 30         sfill0  tsx
07BD 7E   027B  sfill1  jmp   error      no report error

                        * FILESEND MAINLINE *
07C0 0F   02    fsend   clr   filend     clear end of file flag
07C2 01   27            clr   pthnam     clear pathname flag
07C4 86   01            ldaa  #1         open for read access
07C6 BD   0255          jsr   fopen
07C9 8E   0AFC          ldx   #ostq1
07CC BD   CD1E          jsr   putreq     initial NAK message
07CF 8D   0505  fsend1  jsr   fbuff      fill buffer
07D1 8D   0166          jsr   nodin      Dummy read to clear acia
07D4 C6   3C            ldab  #60        one minute timout
07D6 8D   016E  fsend2  jsr   timout     wait for initial NAK
07D8 69   21            tst   rflag
07DA 27   0C            beq   fsend3
07DC 8D   F166          jsr   nodin
07DF 81   15            cmpa  #nak
07E1 26   05            bne   fsend3     if no initial nak - cancel
07E3 BD   05EB          jsr   tfill
07E6 20   03            bra   fsend4     back to menu after normal transfer
07E8 BD   0232  fsend1  jsr   fcls
07EB 39         fsend4  rts

                        * BATCH MODE FILE TRANSFER *
07EC 8E   0AA9  sbatch  ldx   #msg       prompt for filename(s)
07EF 8D   CD1E          jsr   putreq
07F1 BD   CD1B          jsr   inBuf      get name(s) into line buffer
07F4 7F   CC11          clr   lstchr     allow us to read buffer
07F7 86   CC14          ldx   bufpnt     line buffer pointer
07FA 9F   07            stx   xtemp1
07FC 0F   26            clr   wildfl     initialize flags
07FE 0F   27            clr   pthnam
0800 7D   0255  sbtch   tst   alldr      all drives?
0803 27   18            beq   sbat1      no - get name from buffer
0805 8E   0B0F  sbtch1  ldx   #dirfcb
0808 86   10            ldaa  #$10       find drive number
080A A7   04            staa  0,x
080C BD   D40A          jsr   fas        test for errors
080F 26   24            bne   sbat3
0811 9E   07            ldx   xtemp1     restore buffer pointer
0813 9F   CC14          stx   bufpnt
0816 A6   04            ldaa  0,x
0818 97   CC11          staa  lstchr     set last terminator
081A 20   43            bra   sbat8
081C 9E   07    sbat1   ldx   xtemp1     get line buffer pointer
081E A6   00            ldaa  0,x        get first character
0820 81   2E            cmpa  #'.
0822 27   09            beq   sbat2
0824 BD   CD21          jsr   class      classify character
0827 25   04            bcs   sbal2      alphanumeric?
0829 81   39            cmpa  #'9        numeric?
082B 23   17            bls   sbat5
082D 7D   0255  sbal2   tst   loop       looping?
0830 27   05            beq   sbat4
0832 7F   0BCB          clr   bufst
0835 20   50    sbat3   bra   sbat11     send next filename - back to menu
0837 8E   0B0F  sbat4   ldx   #dirfcb
083A C6   0C            ldab  #workdrv    working drive
083C A7   01            staa  1,x        put it in fcb
083E 3A   1F            abx              both drives if $FF
083F 97   25            staa  alldr
0841 20   C0            bra   sbtch1
0843 9E   07    sbat5   ldx   xtemp1
0845 BF   CC14          stx   bufpnt
0848 BD   CD27          jsr   nxtch
084B 84   03            anda  #3         obscure drive number
084D 8B   0B0F          adda  #dirfcb
084F 97   01            staa  1,x        put drive no. in fcb
0851 BD   CD27          jsr   nxtch
0854 25   02            bcs   sbat6      non alphanumeric?
0856 20   3A            bra   sbat14
0858 8E   CC14  sbat6   ldx   bufpnt
085B 9F   07            stx   xtemp1
085D BD   0697  sbat8   jsr   tsttrm     test terminator
0860 26   04            bne   sbat9      not end of line character
0862 0F   24            clr   allflg     clear flag
0864 20   1E            bra   sbat12     try to send file
0866 8E   CC14  sbat9   ldx   bufpnt
0869 A6   04            ldaa  0,x
086B 81   2E            cmpa  #'.
086D 27   10            beq   sbat11     time to get filename
086F BD   CD21          jsr   class      test if alphanumeric
0872 25   1E            bcs   sbat14     no - report error
0874 81   39            cmpa  #'9        numeric?
0876 22   07            bhi   sbat11     no - get name
0878 BD   CD27  sbat10  jsr   nxtch
087B 24   F8            bcc   sbat10     continue til non alphanumeric
087D 20   DE            bra   sbat8      now look for terminator
087F BD   0605  sbat11  jsr   getnam     go get filename
0882 25   0E            bcs   sbat14     test for errors
0884 BD   074E  sbat12  jsr   sfil       send file with Christensen protocol
0887 BD   0697          jsr   tsttrm
088A 7A   F3            dec   wildfl
088C 7E   081A          jmp   sbtch     repeat
088F 7E   06C2  sbat13  jmp   exit
```

```
08AE 7E  06A0    sbatl4  jmp   serr

                 * SELECT SEND MODE *
0  1 80  29      filsen  tst   batch     which mode?
08B3 26  05              bne   filsl
08B5 B9  07D7            jsr   fsend     single file send
08B8 20  03              bra   fils2
08BA BD  08B5    batsl   jsr   batch     batch mode
08BD 0D  28      fils2   tst   gdflg     good transfer?
08BF 27  04              beq   fils3     no - no message
08C1 8E  007F            lds   #msg7     success
08C4 BD  CD1E            jsr   putmsg
08C7 39          fils3   rts

                 ***** EXECUTE FLEX COMMAND *****

08CB 8E  0AB0    dofles  lds   #flexmsg  FLEX prompt
08CE BD  CD1E            jsr   putmsg
08D1 BD  CD1D            jsr   inbuf     get FLEX command in line buffer
08D4 BD  CD40            jsr   docmnd    go do it
08D7 5D                  tstb            test errors
08D8 27  08              beq   dflexl
08DA 8E  0C4F            lds   #fcb       point to fcb
08DD E7  01              stab  1,x        set error code
08DF 7E  0270            jmp   error     report error & back to menu
08E2 39          dflexl  rts

                 **** CANCEL FILE SEND OR RECEIVE ****
08E3 C6  01      cancel  ldab  #1        set 1 second wait for sender to stop
08E5 BD  01AE            jsr   timout
08E8 0D  21              tst   gflag     anything from line?
08EA 27  05              beq   can1      no - cancel filesend
08EC BD  0146            jsr   getin     get char
08EF 20  F2              bra   cancel    throw it away
08F1 C6  03      can1    ldab  #3        send 3 CANs
08F3 86  18              ldaa  #can      cancel
08F5 34  04      can2    psh   b
08F7 BD  01A3            jsr   txchr
08FA 35  04              pul   b
08FC 5A                  decb
08FD 26  F6              bne   can2
08FF 39                  rts
                 **** DISPLAY AND/OR TOGGLE SWITCHES ****
0900 BE  0AE2    setsw   ldx   #swmsge   point to switch names
0903 9F  0F              stx   stloc
0905 9F  07              stx   xtemp1    save it
0907 8E  0029            lds   #stsw      start of switches
090A 9F  05              stx   xtemp
090C 8E  0A0F    setsw1  lds   #swmsg
090F 9F  11              sts   endloc
0911 C6  05              ldab  #5        5 characters in each switch type
0913 BD  0202            jsr   outstrg
0916 9E  05              ldx   xtemp     get switch
0918 A6  84              ldaa  0,x
091A 8E  0AF6            lds   #swmsg4   'ON/OFF' string
091D 4D                  tsta            switch set?
091E 26  06              bne   setsw2    yes - point to 'ON'
0920 30  01              ins
0922 30  01              ins
0924 30  01              ins
0926 9F  0F      setsw2  stx   stloc
0928 BE  0ACD            lds   #swmsgc
092B 9F  11              sts   endloc
092D C6  03              ldab  #3
092F BD  0202            jsr   outstrg   put condition in printline
0932 BE  0A0F            lds   #swmsg    display switch condition
0935 BD  CD1E            jsr   putmsg
0938 BD  CD15            jsr   getch     change switch?
093B 81  0D              cmpa  #cr       no - finish display
093D 27  27              beq   setsw4
093F 84  5F              anda  #$5f      ask to upper case
0941 81  59              cmpa  #'Y       change it?
0943 26  04              bne   setsw3    no - go to next
0945 9E  05              ldx   xtemp
0947 63  84              com   0,x       flip switch
0949 9E  05      setsw3  ldx   xtemp     go to next
094B 30  01              ins
094D 8E  092C            cps   #endsw    last?
0950 27  14              beq   setsw4    yes - back to menu
0952 9F  05              sts   xtemp
0954 9E  07              lds   xtemp1
0956 31  01              ins
0958 31  01              ins
095A 31  01              ins
095C 31  01              ins
095E 9F  07              stx   xtemp1
0960 9F  0F              stx   stloc
0962 20  A6              bra   setsw1
0964 39          setsw4  rts

                 ***** COMMAND TABLE *****

                 09A4 cmdtbl  equ   *
0764 43                  fcb   'C
0765 08E0                fdb   cancel
0767 52                  fcb   'R
0768 024F                fdb   getfil
076A 52                  fcb   'R
076B 04AB                fdb   filrec
076D 53                  fcb   'S
076E 0001                fdb   filsen
0770 54                  fcb   'T
0771 0570                fdb   term
0773 55                  fcb   'U
0774 0BC0                fdb   dofles
0776 5A                  fcb   'Z
0777 08F0                fdb   setsw
0779 04                  fcb   4

                 ***** STRINGS AND MESSAGES *****

077A 0C          signon  fcb   ff        form feed
077B 20 20 20 20 20              fcc   '                    Modem68 by John Moorfoot'
0787 0A 0D               fcb   $0a,$0d
0789 20 20 20 20 20              fcc   '                         Version 1.0 Aug/83'
0795 0A 0D 0A 0A         fcb   $0a,$0a,$0a,$0a,$0a,$0d
                 07B7 pmtmsg  equ   *
07B7 20 20 43 20                 fcc   '  C  - Cancel file transmission or reception'
07BB 0A 0D               fcb   $0a,$0d
07BD 20 20 52 20                 fcc   '  R  - Return to files'
07C1 0A 0D               fcb   $0a,$0d
07C3 20 20 52 20                 fcc   '  R  - Receive file'
07C7 0A 0D               fcb   $0a,$0d
07C9 20 20 53 20                 fcc   '  S  - Send file'
07CD 0A 0D               fcb   $0a,$0d
07CF 20 20 54 20                 fcc   '  T  - Terminal mode'
07D3 0A 0D               fcb   $0a,$0d
07D5 20 20 55 20                 fcc   '  U  - Execute FLEX command'
07D9 0A 0D               fcb   $0a,$0d
07DB 20 20 5A 20                 fcc   '  Z  - Display/Set switches'
07DF 0A 0D               fcb   $0a,$0d
07E1 53 65 6C 65                 fcc   'Select option '
07E5 04                  fcb   4
07E6 45 6E 74 65 rmsg    fcc   'Enter file name'
07F4 0A 0D 04            fcb   $0a,$0d,4
07F7 20 20 20    fnmsg   fcc   '   '
07FA 04                  fcb   4

07FB             swmsga  rmb   5
07FE 20 23 77 69 swmsgb  fcc   ' switch #'
0803             swmsgc  rmb   3
0806 20 20 20 63 swmsgd  fcc   '  - change (Y/N)? '
0816 04                  fcb   4
0817 42 61 74 63 swmsge  fcc   'BatchLog  Echo  CRCC '
0827 20 4F 4E 4F swmsgf  fcc   ' ONOFF'

                 * FILE TRANSMISSION MESSAGES *

082B 61 77 61 69 msg1    fcc   'awaiting initial nak'
083F 0A 0D 04            fcb   $0a,$0d,4
0842 74 69 6D 65 msg2    fcc   'timeout #'
084D 04                  fcb   4
084E 72 65 63 65 msg3    fcc   'received duplicate sector #'
0869             msg3a   rmb   1
086A 0A 0D 04            fcb   $0a,$0d,4
086D 73 65 6E 64 msg4    fcc   'sender CANcelled'
0880 0A 0D 04            fcb   $0a,$0d,4
0883 72 65 63 65 msg5    fcc   'receiver CANcelled'
0896 0A 0D 04            fcb   $0a,$0d,4
0899 6E 6F 20 41 msg6    fcc   'no ACK on sector: abort'
08B7 0A 0D 04            fcb   $0a,$0d,4
08BA 66 69 6C 65 msg7    fcc   'file transfer is successful'
08D9 0A 0D 04            fcb   $0a,$0d,4
08DD             msg8a   rmb   8
08E5 2E          msg8b   fcc   '.'
08A6             msg8c   rmb   3
08A9 00          msg8d   fcb   0
08AA 20          msg8e   fcc   ' '
08AB             msg8f   rmb   7
08B2 0A 0D 04    msg9    fcb   $0a,$0d,4
08B5 4F 70 65 6E         fcc   'opened closed created'

                 ***** FILE BUFFER *****

08CA 01          soh     fcb   1         block start character
08CB             blkcnt  rmb   1
08CC             blkchk  rmb   1
08CD             bufst   rmb   128
                 0E4C bufend  equ   4-1
0E4D             chblk   rmb   2

                 ***** FLEX FILE CONTROL BLOCKS *****

                 0C4F fcb     equ   *

                 0D0F dirtch  equ   0+320

                 0F33 stack   equ   0e40     program stack

                              end   start
                              end   start

0 ERRORIS DETECTED

SYMBOL TABLE:

abtflg 0026  aciac  E00A  aciad  E00B  ack    0006  alldr  0025
allflg 0024  astk   002A  base   C000  batch  0029  blkchk 08C
blkaum 08C0  bufend 0C4C  bufpot CC14  bufptr 0000  bufst  08CD
build  04CA  build1 0480  build2 04F1  build3 0507  build4 04FC
can    0010  can1   08EE  can2   08F2  cancel 08E0  canflg 0022
check  0180  chblk  0C4D  chsum  0003  class  CD21  cmdtbl 09A4
cr     000D  crcalc 05EC  crcc   002C  crchk  02F5  ctlc   0005
ctlc   001A  dflex1 08BF  dirtch 000F  docmnd CD40  dofles 08CB
echo   0020  endblk 001E  endloc 0011  endsw  002C  eot    CC07
eot    0004  error  0270  errsp  027D  exit   04C2  ext    0010
fbuf25 05A9  fbuf   0585  fbuf11 058A  fbuf12 0581  fbuf13 05B1
fbuf14 05BF  fbuf15 05E0  fcb    0C4F  fcls   0232  ff     000C
fileq  0021  filend 0002  filrl  0401  filr2  040C  filr3  040F
filr4  04C9  filrec 04A8  filsl  08BA  fils2  08BD  fils3  08C7
filsen 08B1  first  0318  firstl 031A  firstl2 032C  firstl3 032E
flnmsg 0A0B  los    0406  fpscln 0403  fsmg   0AA9  fopen  0255
fopen1 0276  fopen2 027A  frcvl5 0350  frcv32 0305  frcv35 0392
frcv37 03BB  frecv  0348  frecv8 0354  frecv1 0359  frecv2 035F
frecv3 0376  frecv4 038D  frecv5 03C4  frecv6 03D0  frecv7 03E7
fsend  07D7  fsend1 07E6  fsend2 07EE  fsend3 0801  fsend4 0804
gdflg  0028  getcl  011C  getc2  012A  getc3  013F  getch  CD15
getcmd 0114  getfil CD20  getin  06CB  getma2 0A0C  getma1 06F1
getma5 06F9  getma6 0700  getma7 070B  getmae 06B5  getfle 024F
inbuf  CD1B  iost   0147  kbd    E008  kbdin  01A0  kbdst  0150
lmg    0537  lmgl   0544  lmg2   0561  lmg3   056E  lmgg   0073
lstirn 0C11  modin  0260  modist 0150  modout 0152  outbuf 0147
movmem 065C  movmv1 069D  movmv2 06A7  msg1   04FC  msg2   0813
msg2a  081C  msg3   081E  msg3a  0839  msg4   0830  msg5   0850
msg6   0865  msg7   087F  msg8a  090D  msg8b  08A5  msg8c  08AA
msg8d  08A9  msg8e  08AA  msg8f  08AB  msg9   08B5  mvlmem 0290
mvstrg 0202  nak    0015  name   0013  nrst   032F  nxtch  0345
next2  0347  null   0000  outbuf CD27  outdec CD3C  pcrlf  CD24
putmsg 09D7  prnt2  01B0  prnt3  01F3  putrng CD1E  pthmem 0027
purge  01C1  purge1 01CF  purge2 01D7  purge3 01D9  putchr CD10
rbal1  03F5  rbal10 04A4  rbal11 04A5  rbal2  03FB  rbat3  0414
rbat4  041E  rbat5  0425  rbat6  0427  rbat7  0430  rbat8  040F
rbat8a 040C  rbat8b 0473  rbat9  0478  rbat9a 048A  rbatch 03E8
rblok  02B0  rblok1 02C7  rblok2 02CF  rblok3 02E3  rcvblk 0020
rpterr CD3F  rsch   02E3  rschl  02F1  rsch2  02F4  rschr  016A
rxchr1 01AE  sbat1  0036  sbatl0 0094  sbatl1 0098  sbatl2 00A0
sbatl3 00A8  sbatl4 00AE  sbat2  0047  sbat3  004E  sbat4  0058
sbat5  005E  sbat6  0074  sbat8  0079  sbat9  0082  sbatch 0095
sblok  05C3  sblok1 05C8  sblok2 0587  sblok3 058E  sbtch  0014
sblchl 001E  scrlog 002A  send   070E  send1  0718  send2  0724
send3  072C  send4  073F  send5  0740  serr   06A0  setext CD33
setsw  09F0  setsw1 0909  setsw2 0923  setsw3 0946  setsw4 0963
setxl  02F6  setxtl 0306  sfil   074E  sfill  075C  sfill0 0783
slill1 07B4  sfil2  075E  sfil3  0766  sfil4  0770  sfil5  0780
sfil6  0196  sfil7  079C  sfil8  07A0  sfil9  07B5  signon 077A
```

HAZELWOOD DM-256 MEMORY SYSTEM REVIEW

HAZELWOOD DM-256 MEMORY SYSTE

About this time a year ago we published a review of the Hazelwood DM-64 Memory System. It was a good review, the memory board is a good system. It was just that simple.

For the past couple of years I have used the DM-64 in my personal computer and it has never dropped a bit, or even hinted a cough. Even at system speeds in the range of 2 mhz it works. Which should dispel the old assumption that dynamic memory has some sort of problem.

In the early, early days some dynamic memory did exhibit quirks, so did some static. However, most of the problems, in the beginning was due to poor engineering practices on the part of hardware design types. They didn't know how to interface to dynamic devices. Granted it is simpler to design to a static system rather than a dynamic, but once done properly there should be little of any difference in thru-put of the host system. Hazelwood's Mike Smith has done his homework well, he knows how to design to dynamic standards. His (Hazelwood) memory systems work.

DM-256KB Memory Board description

As the name implies the system is a 256 kilobyte dynamic memory system, engineered to one board. Operating system speeds up to 2.5 mhz are guaranteed with NO clock stretching or cycle stealing. As pioneered on the Hazelwood 64KB board this 256KB system also utilizes "High Speed Transparent Refresh" a technique developed by HAzelwood.

As delivered the system can be utilized on either the Hazelwood 64 pin backplane or the Standard S50 Bus backplane equally well. The only system that may present some problem for this memory system is the GIMIX IIC system, due to hardware consideration of the GIMIX system. On all other computer systems we have used this memory system with, including FLEX™, OS9™, UniFLEX™ and a couple other non-standard, it worked flawlessly.

With the SS50 Bus there is 20 bit addressing, and 4 additional bits of addressing available on the SS-64 Bus.

The system may be programmed by dip-switches to respond as 4 64KB boards or as 16 16KB boards. By this arrangement total access is afforded to practically all current operating systems.

To quote from the specifications sheets for the DM-256:

"The refresh control on the DM-256 uses odd/even interleaving to optimize the charge on each memory cell."

What this means is simply this. While an even address is on the bus all of the odd cells are refreshed and while an odd address appears on the bus the even cells are refreshed. In addition all cells receive refresh burst when there is a non-VMA cycle. Simple!

The net result of this is that there is an actual "over refreshing" by common standards, but it insures that cell charges are held to a maximum value over the entire refresh cycle and to the next.

Many other dynamic memory designs refresh on the non-VMA cycle only, to our knowledge only Hazelwood dynamic memory systems enjoy the benefits of "over-refresh". The guarantee is for operation up to 2.5 mhz system speed (on the standard SS50 or SS64 system this would require a 10 mhz xtal). Boiled down this means that whereas most other dynamic systems using only non-VMA refresh are prone to "soft" memory errors. And depends to a large extent on the particular program being run on the computer, and the amount of non-VMA activity. Hazelwood claims no program-dependency due to their even/odd address refresh. This consistent distribution of odd/even addresses appearing on the bus, in addition to the conventional non-VMA insure "hard" memory data retention.

Hazelwood's documentation cites laboratory testing at 2.5 mhz with 250 ns memory chips. Normally it is necessary to use 200 ns dynamic devices to achieve a 2.5 mhz speed. This would be marginal. However, to insure "sweet" data, 150 ns memory chips or better are installed on the DM-256 KB memory system, as delivered.

In addition to the above this system does not begin its read or write cycle until the bus signal "E" goes from high to low (mid-cycle), allowing maximum time for the dynamic address translator (commonly referred to as the DAT, a portion of the ROM system firmware) to develope the effective bus address. Some static designs are detracted linearly from access time margins by DAT activity.

A nice feature of the entire system is that there is no complicated timing of any sort, and no hard to o tain or exotic parts are used on the board.

Set-Up

The DM-256 has several options:

The board may be set to respond or not respond to all EXXX or FXXX addresses or only to those in the high 64K bank.

Addressing for the SS50 Bus to 1MB and to 16 MB on the Hazelwood S64 Bus.

There are wo jumpers "E" - "F":

"E" The E jumper controls response in the XEXXX range. The three (3) possible responses are:
1. Alway - not normally used due to address conflicts with the I/O structure.
2. Never - used for non-extended addressing.
3. Respond to all except "F" - gives maximum available memory to extended memory systems.

"F" The F jumper controls response in the XFXXX range. The three (3) possible responses are:
1. Always - not normally used because of address conflicts with the CPU card.
2. Never - see 2 above.
3. Respond to all except "F" - see 3 above.

An 8 pole switch controls board selection:

1,2 determines whether the board appears as the first, second, third or fourth 256KB board in the system. 1=LS bit 2=MS bit. ON= binary 0 - OFF= binary 1.

3-6 for addresses beyond 1 megabyte range and could only be used with the Hazelwood "HELIX" 64 pin system which can address up to 16 megaby es of memory. Presently set to OFF.

7 disables response to address line A19 and is normally ON.

8 selects contiguous or non-contiguous mode of addressing.

Conclusion

The system comes assembled, tested and burned in. Gold bus connectors insure stable trouble-free operation. And is backed up by a one year factory warranty.

Hazelwood Computer Systems, Inc.
907 East Terra
O'Fallon, MO 63366
314 281-1055

PRICE $995.00 suggested factory price

The Hazelwood 256K dynamic system may be ordered direct from the factory or CALL DATA-COMP 1-800-338-6800 for delivery end pricing information.

- - -

Editor's Note: Coming soon will be reviews of additional Hazelwood and HELIX™ systems and components. Also we hope to have soon a review of the new SWTPC desktop computer, as well as other computers and systems. All because you our readers have asked us to. In addition will be some 68000 systems, some with software you know and some you don't. Fact is as this is being written,

two of our staff are out of town collecting information on 68000 systems. Our new review policy began with a review of the GIMIX III system last month. These will be inhouse reviews, as opposed to readers reviews.

We still encourage reader reviews, however, things are beginning to move in varied directions and there will be more inhouse reviews presented in the coming months. Don't be surprised at we broaden into the 68000 systems as well. **And if you have anything on ANY 68000 systems, hardware or software, please let me know. Our readers** are beginning to show a lot more 68000 interest since the advent of some fairly decent 68000 applications software, and nearing release of more 68000 devices for the Standard S50 Bus. Many at very good prices. Low enough in cost to appeal to the hobbyist, remember them, they were the guys (and some gals) who got it all started. I was one of them, still am, proud of it and getting ready to get my "feet wet" with a semi-kit 68000 system. Join me, I'm gonna have some fun again!

There is one thing that I want to make very clear; **we (Computer Publishing and 68 Micro Journal) will not forget or abandon our FLEX™, UniFLEX™, SSB™, OS9™, etc. readers.** I have heard your complaints of non-support or total abandonment of our group, by some of the originators in our group. I can assure you that it has come back home to roost for some. Their reputations suffered. (i.e. pocketbooks and business volume also) I know of two in particular who spoke from both sides of their mouths. They didn't reckon the power of user satisfaction, or word of mouth reputation. They kept promising to support what they had sold us while all the time devoting effort that should have been put to work getting the stuff they sold us bug-free, developing other markets, with a "to hell" attitude towards us. I know for an absolute fact it has cost them dearly!!! Funny, but you would have thought that they would have suspected that someone interested in their products might just ask one of us. You would be deeply surprised at the feedback I get.

As some of you might know, I have not been too active recently. Back on November 28th last year, I underwent open-heart surgery and five bypasses. Since then I have not been in the office much, but am feeling better each day, so expect to be more active - soon! To all of you who sent flowers, cards, fruit, letters, telephone calls, etc. THANK YOU! And PLEASE keep in touch! GOD BLESS!!

DMW

- - -

MULTI-TASKING WITHOUT INTERRUPTS

MULTI-TASKING WITHOUT INTERRUPTS

by **Graham Trott** (author of PL/9)

Have you ever written a program that does two or more jobs at the same time? If you have, then you've probably run across the problem of resource allocation, going something like this:

The main program is cycling round, looking at the keyboard to see whether you want it to do anything. When you hit a key it goes into a flurry of activity while it sorts out your request. If this activity only means updating some variables or doing a few calculations, then the program will get back to the keyboard before your finger has even released the key. But suppose that you've asked it to set in motion some complex I/O function, such as ramping up the voltage on a D/A converter, which requires the program to change a value every time a timer overflows? In a single-task program, your keyboard will go dead until that task has finished. The problem is compounded if there are several D/A converters, each ramping at different rates.

OK, so it's possible to write a single-task program that can handle this sort of job. The problem then is that each of the tasks gets thoroughly bound up with each of the others, to the extent that maintaining the program becomes a real pain. What would be really nice would be to write each of the tasks separately and tell the CPU to execute all of them in parallel.

Now I know that OS-9 is out there just waiting to be used for this sort of job. OS-9 is great if the drivers already exist for the I/O device you have to handle, but the job of writing special interface drivers is just too much for simple hackers. Furthermore, you don't actually know when OS-9 is going to suspend a task and start up another, which makes interlocking variables a bit tricky. However, if you can handle OS-9 and if OS-9 can handle your application, fine.

If you're still here then you must be a FLEX hacker like me. (Either that or you're an OS-9 nut with a streak of curiosity, waiting for me to make a fool of myself. No matter.) What I'm going to describe is a simple technique for writing programs that are divided into separate tasks, all operating concurrently, with as much or as little interaction between them as you wish, and all without the use of interrupts.

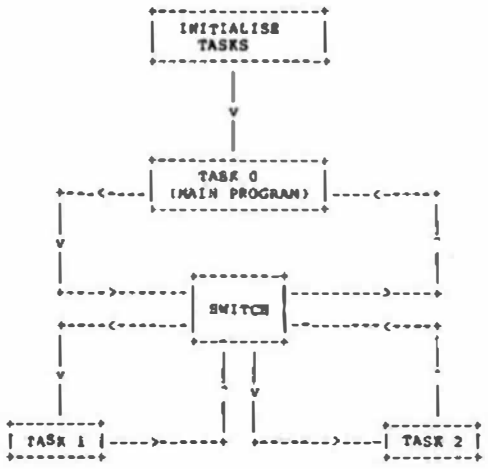
The heart of the technique is a routine called SWITCH, whose job it is to change from one task to another. Suppose that three tasks are set up; a main program and two secondary jobs called TASK1 and TASK2. Each must contain sufficient calls to SWITCH to avoid hogging the CPU. When SWITCH is called, if the main program is running then it is suspended and TASK1 is resumed; if TASK1 is running then TASK2 is resumed and if TASK2 is running then the main program is resumed. Each task can run for as long as it likes, then a call to SWITCH gives the next task some CPU time in a round-robin fashion.

The advantage of this scheme is that unlike one that uses timer interrupts there is no need to guard against one process interrupting another and interfering with a variable that is being worked with. If you want to mess with a variable you do it, only handing control over to the next task when you're good and ready.

The way the system works is quite simple. As part of your program's scratch space you allocate a stack for each of the secondary tasks. The starting address of each secondary task is placed at the very top of its reserved stack area. A two-byte pointer to this stored address is also kept for each task and one is reserved for the main program. Lastly, a byte is reserved that records which task is currently active. This byte is initially set to zero, the main program.

Having done this initialisation, the main program can now start. At some point it is necessary to start up the multi-tasking system. A good place to put a call to SWITCH is in the program loop that polls the keyboard to see if a key has been pressed. This ensures that as long as no key is operated a steady supply of calls to SWITCH will be generated.

The first call to SWITCH causes the stack pointer to be saved in the two-byte pointer reserved for the main program. It is then re-loaded with the contents of task 1's pointer and a return-from-subroutine is executed. This causes the address of task 1 to be put into the program counter, which starts it executing. At some point it too calls SWITCH which in the same way saves the current status and starts up task 2. When task 2 calls SWITCH the main program is resumed at the point it left off. This can be shown clearly by diagram:



Note the most important feature of the diagram, that once the multi-tasking has begun it goes on forever. None of the tasks can be allowed to finish or the system will stop. In practice, this restriction can be overcome by allocating flags that tell the SWITCH procedure which tasks are currently active. In this way, new tasks can be introduced when they are needed. The alternative is for a inactive task to stay in an endless loop, generating calls to SWITCH.

The following PL/9 program should illustrate the use of this technique. Firstly, the necessary scratch variables. These are declared here as GLOBAL variables but they could equally well be declared using the AT statement. Since this program will actually run, I'll put ORIGIN and STACK statements first:

```
/* Multi-Tasking Demonstration Program */

origin=$8000; stack=*;   /* i.e. stack=$8000 */

constant no_of_tasks=3;     /* Tasks 0, 1 and 2 */

global

   byte dummy1(100):  /* Length of stack for task 1 */
   integer stack1:

   byte dummy2(100):  /* Length of stack for task 2 */
   integer stack2:

   integer task_table(no_of_tasks):
   byte task;
```

The user's own variables can be included anywhere in this list. Note that a pointer in "task_table" is provided for each of the tasks, including the main program. Each of the secondary tasks has a stack area (variables "dummy1" and "dummy2") and space for the return address (variables "stack1" and stack2"). The main program's stack is the rest of the system RAM so there's no need to allocate a separate stack. The byte variable "task" records which task is currently active.

The SWITCH procedure must be near the start of the program, since it's called by other procedures (PL/9 requires that procedures and variables are declared before they are called). It looks like this:

```
Procedure switch;
   xreg=.task_table(task);
   gen $10,$ef,$84;        /* STS 0,X */
   task=task+1;
   if task=no_of_tasks then task=0;
   accd=task_table(task);
   gen $1f,$04;            /* TFR D,S */
endproc;                   /* Do a "RTS" */
```

Now comes the body of the program, including the secondary tasks. The first of these is called task_1 (but you can use any name you wish). Its job is to cause the terminal bell to ring at regular intervals. Note the call to SWITCH in the inner loop:

```
procedure task_1: integer count;
   repeat
      count=1000;
      repeat
         switch;
         count=count-1;
      until count=0;
      acca=7;              /* ASCII "BELL" */
      call $cd18;          /* FLEX "PUTCHR" */
   forever;
endproc;
```

The second task, called task_2, sends to the terminal the complete ASCII character set from $20 through $7D, over and over again:

```
procedure task_2: byte char;
   repeat
      char=$20;
      repeat
         switch;
         acca=char;
         call $cd18;
         char=char+1;
      until char=$7e;
   forever;
endproc;
```

The next routine forms part of the main program. It echoes to the printer every key that is typed. It is so designed as to generate calls to SWITCH while it is waiting for a key:

```
procedure echo_to_printer: byte char;
   call $ccc0;                /* Initialise printer */
   repeat
      repeat
         switch;
         call $cd4e;          /* FLEX "STAT" */
      until (ccr and 4)=0;
      call $cd15;             /* FLEX "GETCHR" */
      char=acca;
      call $cce4;             /* Output to printer */
   until char=$1b;            /* Wait for <ESCAPE> */
endproc;
```

Lastly we have the main program, which as its first job must set up the multi-tasking system:

```
procedure main;

   task=0;

   task_table(1)=.stack1;     /* Address of stack1 ->
                                 table */
   stack1=.task_1;            /* Address of task 1 ->
                                 stack1 */

   task_table(2)=.stack2;     /* Address of stack2 ->
                                 table */
   stack2=.task_2;            /* Address of task 2 ->
                                 stack2 */

   echo_to_printer;
```

All that "main" does after initialising the multi-tasking system is to call "echo_to_printer" to run the primary task.

If you were expecting a serious example I'm sorry to have disappointed you. A multi-tasking system of this kind is designed to cope with real-time applications where actions are taken dependent upon some outside event, and it's rather difficult to produce a realistic

example when the only outside event is the keyboard. You can satisfy yourself that the program works, however, providing you've got a copy of the PL/9 compiler. Otherwise it is a fairly simple task to translate the above into assembly language or the high-level language of your choice (but be warned, many languages will find stack manipulations such as are used here difficult to swallow).

REPAIRING MINI-FLOPPY DISK DRIVES

Francis MASSEN
8 Cite Strauss
L-9357 BETTENDORF/Luxembourg, Europe
tel: 808021

==========
REPAIRING MINI-FLOPPY DIS DRIVES
==========

Usually we do not like fiddling with our disk drives, preferring to ignore that these are the most error-prone parts of our computer system. But when DIS READ ERRORS begin to appear daily, and when swapping disks written by another system becomes a pain, then there is no issue: demount your drive and recalibrate it!

When I remember the troubles I have had with floppy-drives, they can be classified as follows:

1. recalibration needed
2. mechanical troubles
3. electronic faults

This list gives the true frequency of errors creeping up: most problems boil down to a decalibration, some to a mechanical fault and very few to an electronic failure. Let me discuss in that order these problems and tell you what can be done, without being an electronics professional or holding a degree in mechanical sciences.

1.Recalibration

To recalibrate a drive, you first need a calibrating diskette (for instance VERBATIM DS5-SDI-CE); these are expensive but you cannot do a proper work without . The steps to follow are described in detail on the enclosure of the diskette, and you also find plenty of hints in the operation manual of your drives manufacturer. Another must is a dual-trace oscilloscope.
From personal experience I can say that often you only need to recalibrate track 16 (look at the cat-eyes pattern) and the track 0 sensor (a micro-switch or a photo-transistor). Programs helping you to position the head on one of the available tracks have been published in MICRO68; for instance I use the DISKEDIT program written by R.P.Lajeunesse (MICRO68 Sept. 1979): it's a very useful old goodie! You may have to modify it to your maximum track number, but that's no problem.
If you do not have a calibration diskette nor an oscilloscope there is a last-hope method: load a disk-inspector program such as TSC EXAMINE and try to read informations from every track, adjusting the stepper-motor by rotating it gently after *having* loosened the 2 or 4 screws who fix it. If you get no more read-errors for neither track, you may be somewhat confidant that your drive is recalibrated to the diskette you are using. If that diskette is a copy made at a time where your drive was still ok, that may do it; if not, ales, using a calibration diskette is a must!

Not every manufacturer makes his drives with the same care as others do; MPI drives are notorious for their unreliability; if you have to sell a computer system to a non-technical minded customer, do not give him MPI drives! As a result, many customers are swapping their MPI drives to SHUG RT for instance, so that you may have the opportunity to buy very cheaply an old MPI. The most error-prones seem to be the double-headed , 40 tracks 52-models. Often only one side may be used, the second giving you plenty of formatting errors or refusing formatting at all. Do not be afraid: these drives may be cured!

Begin to control the rotational speed of the disk motor, looking at the stroboscope disk-pattern; then recalibrate track 16 and control track 0 alignment, eventually track 39.
Looking at the electronic boards you see 2 female connectors coming from the 2 heads; remove one, and try to format the disk in single-sided mode. If you get a crash at the verifying procedure, you removed the wrong head. Swapping the connectors (pay attention at their orientations!) gives you a clue that both are working.

On most drives you notice that the second head may be slightly moved independly from the first one; I found that fiddling with the skew of the second head can solve the problem (for instance: if you are unable to read a long catalog on a double-sided disk, who spreads over both sides, try the adjustments until you may read the whole catalog: now you are sure that both heads are working and may only need some minor alignments.)

Recalibrating a drive requires time and patience, but it pays off! There is nothing more rewarding than to read a "1404 sectors free" message on a drive that always gave you a "formatting aborted". It is a good idea to set up a workbench with a separate power supply for your defect drive; I included a Y-toggle switch between my controller and the normal drives, so that flipping a multiple-pole switch connects the controller to the drive in repair.

2.Mechanical faults

Here are the most frequent faults I found:
* the guiding rods of the head assembly give too much friction; in theory these rods *ought* not to be oiled , but in practice a droplet of paraffine-oil often will do miracles.

* the latest MPI drive I repaired behaved in a strange manner: formatting a new diskette sometimes worked well, then at other times nothing could be done. Careful inspection showed that the principal head (that mounted on a hexagonal base) had broken loose. This head usually is glued to the carriage with some sort of two-component glue; in my case, a barely visible creck showed that the glue was broken, so that sometimes the head moved slightly from its normal position. One drop of standard two-component glue fixed the problem!

* the most gruesome mechanical fault I found was also in a MPI model 52 drive. That factory-new drive had passed the MPI quality controls, but refused absolutely to work. The disk motor rotated well, but the stepper motor was tightly blocked.
I disassembled the whole hardware, opened the stepper-motor and found that it COULD not rotate: the dia ter of the rotor was too large, so that the dents of the magnetic rotor gripped those of the stator; the whole motor-case was filled with magnetic chips ! This (Japan-made) motor never could have made a single turn ! How this piece of equipment passed the various inspections and quality controls remains sort of a mystery. (**)
At our school, we gently removed with a lathe a few fractions of a millimeter from the rotor, and now that drives works with a delicate "snurr"; it is my best-sounding and best-working drive! So do not be afraid to open a stepper-motor, even if you never have done it before. First it is instructive, and second it contains very few parts, so that dissamble and reassembling is quite easy.

3. Electronic faults

As a rule of thumb you can be sure that the odds that your non-working drive has an electronic failure are 1 to 99 . In 3 years I only found one fault on the electronics board: that was a 35 track SHUGART drive whose disk-motor suddenly began to rotate, even if the drives were not accessed by the controller (no motor-on). The disk-motor is operates through a current-driver transistor: it's the only big transistor on the board. Shugart uses a TIP140 transistor; replacing it cured the problem. On some MPI models, that transistor runs very hot, as it is only marginally cooled by a little piece of aluminum; so if you notice a motor who starts moving when he ought not (usually at a very low speed) first have a look at that transistor.

Conclusions:

From all what I wrote, you may see that repairing a drive requires some time end patience; that's the main reason for some big vendors not to repair at all a faulty drive, but to replace it by a new one and to scrap the non-working drive.
For you as a hobbyist you here have an opportunity to

FOR THE ELEKTRA

- **OS-9** with Editor, Assembler, and Debugger **$250.00**
- **STAR-DOS Level 1** (FLEX™ compatible but with up to 10 active drives: i.e. 4 8" floppies, 4 5" floppies, and 2 Winchester drives)
  Regular price **$75.00** Introductory price **$50.00**
  (Adaptation Guide $25)

FOR OS-9 by Epstein Associates

- **Super Modem Program** with autodial, configuration file, etc.
  (Available exclusively through AAA Chicago Computer Center) **$100.00**
- Coming — Complete software for Western Union EasyLink

THE ELEKTRA SUPER FLOPPY CONTROLLER

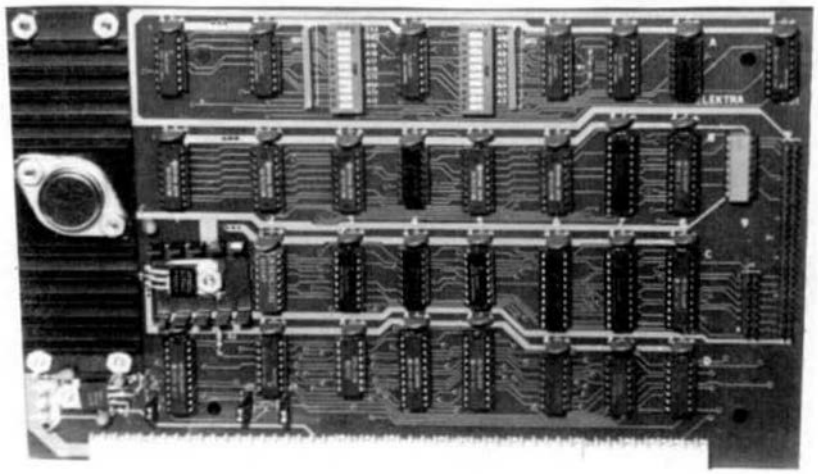
- Emulates the DC-1, DC-2, DC-3, DC-4 as well as the GIMIX #28, #38, #48, and #58 controllers **$275.00**

THE ELEKTRA WINCHESTER SYSTEMS with up to 11 bit burst error correction

New	7 Megabyte Single Drive System **$1995.00**	14 Megabyte Dual Drive System **$2995.00**
•	12 Megabyte Single Drive System **$2295.00**	24 Megabyte Dual Drive System **$3595.00**
•	19 Megabyte Single Drive System **$2995.00**	38 Megabyte Dual Drive System **$4695.00**
New	Winchester Electronics (no drives)	**$995.00**
New	DMA SS-50C Bus Interface (No controller nor drives)	**$695.00**

THE ELEKTRA DBI BOARD

(DMA SS-50C Bus Interface for the ELEKTRA Winchester Systems)



Phone:
AAA Chicago Computer Center
Technical Consultation available most weekdays from 4 p.m. to 6 p.m. CST
(312) 459-0450 120 Chestnut Lane Wheeling, IL 60090

See our catalog and ordering information on the next page.

HELIX

64K 6809 Computer	$2395.00	64K 68008 Computer	$2495.00
256K 6809 Computer	2895.00	256K 68008 Computer	2995.00

Other sy tems available

20 Megabyte 5" add on Winchester System	$2595.00
64K CMOS Static memory board with battery backup	385.00
DMA 5" and 8" Floppy Controller with Dual on Winchester controller I/O	695.00
DMA 5" and 8" Floppy Controller $495.00 6809 CPU Board	495.00
68008 board for SS-50 585.00 CP/M-68K	350.00

Need FLEX, UniFLEX, OS-9 Level I, or OS-9 Level II? We have a system for you!

ELEKTRA COMPUTER CABINET THE LARGEST SS-50 COMPUTER CABINET AVAILABLE! Made of heavyweight 0.090" thick aluminum. Interior is 18-1/2" wide by 21-7/8" deep by 6-3/4" high. Heavy duty A.C. line cord. A.C. fuse holder. EMI filter. Fan with filter. Back panel has 10 cutouts for 'D' type data connectors. Front panel has key on/off power switch, 2 illuminated push button switches (Reset and NMI/Abort), and two cutouts for 5-1/4" disk drives. $250.00

Filler Plate for 5-1/4" drive opening: $10.00 Fan Filter: $10.00

POWER SUPPLY Premium quality linear power supply CONSERVATIVELY rated at 15a @ 5v, 3a @ 16v, 3a @ -16v. Multi-tapped primary for line tuning. $200.00

DISK REGULATOR BOARD WITH CABLES Standard version for 2 floppy drives $50.00
Heavy duty version for 1 Winchester drive and 1 floppy drive. $75.00

AUXILIARY POWER SUPPLY to power second Winchester drive $125.00

ELEKTRA UNIVERSAL SS-50/SS-50C MOTHERBOARD Heavyweight 0.125" thick, 18" long by 9" wide. 11 memory (50 pin) slots. 8 I/O (30 pin) slots. Complete address decoding and selection, as well as extended address capability, for I/O slots. Choice of 4, 8, or 16 addresses per I/O slot. 1" spacing between all memory and I/O slots. On board baud rate generator with low and high ranges providing jumper selectable rates of 75 through 38,400 for each of the five baud rate lines. Allow device circuitry permitting 1 MHz 30 pin disk controllers to run with 2MHz 50 pin CPU boards.

Mounting hardware	$5.00	Bareboard w/documentation	$80.00
Kit w/gold connectors	$400.00	Assembled w/gold connectors	$460.00
Kit w/tin connectors	$320.00	Assembled w/tin connectors	$380.00

ELEKTRA CHASSIS Includes cabinet, 110v power supply, power supply cables, standard disk regulator board with power cables, motherboard with gold square pin connectors, assembled and tested. $950.00

ELEKTRA CPU 8/9 Use either the 6802 or 6808 (to run 6800 software) or 6809. Has provision for up to 3 2716 Eproms, 1K scratchpad, 6840 triple timer, and an optional baud rate generator providing baud rates from 110 through 38,400 baud in two user selectable ranges. Run OS-9, FLEX™, STAR-DOS.

Bareboard:	$50.00	Kit: $225.00	Assembled: $275.00

Optional Baud Rate Generator. $25.00

ELEKTRA DPS DUAL PORT SERIAL CARD Fits the standard 30 pin SS-50 bus I/O slot. Can be configured for 4 or 16 addresses per port. RTS, CTS, DTR, DCD, tRO, FIRQ/NMI, and baud rate can be appropriately implemented for each port.

Bareboard:	$20.00	Kit: $75.00	Assembled: $95.00
Cable (two needed per board)			Each: $20.00

ELEKTRA DPP DUAL PORT PARALLEL CARD Fits the standard 30 pin SS-50 bus I/O slot. Can be configured for 4 or 16 addresses per I/O slot. The direction of the TTL buffers can be controlled by either on board jumper connectors or by a signal from the peripherals. The interrupt request line for each port may be individually jumpered to either the IRQ or FIRQ/NMI bus line.

Bareboard:	$20.00	Kit: $60.00	Assembled: $80.00
Cable (two needed per board)			Each: $20.00

ELEKTRA 64K STATIC RAM/ROM MEMORY BOARDS with gold connector (tin available) Assembled and tested: With 56K RAM $269.00 With 64K RAM $299.00
Kit With 56K RAM $219.00 With 64K RAM $249.00

ELEKTRA UNIVERSAL SUPER FLOPPY CONTROLLER THE BEST 30 PIN FLOPPY DISK CONTROLLER THAT YOU CAN BUY! Controls up to four 5-1/4" drives and four 8" drives for a total of eight system drives. (FLEX system limit is four drives.) Single density or double density, 1MHz or 2MHz, 6800 or 6809. (Double density 8" must be at 2MHz; all other combinations of performance are possible.) Analog phase locked loop data separators with separate adjustments for 5" and 8" drives. Analog write precompensation circuit with separate adjustments for 5" and 8" drives. Designed to meet the data hold requirements of Western Digital floppy controller IC. Assembled and tested. $ 75.00
Disk with drivers, setup, and formatting routines. Specify 6800 (FLEX or Gen. FLEX), or 6809 (FLEX9 or Gen. FLEX), or OS-9, 5" or 8". 30.00
STAR-DOS (Specify 5" or 8") Regular price $75.00 Introductory price 50.00

ELEKTRA WINCHESTER SYSTEMS THE BEST WINCHESTER SYSTEMS THAT YOU CAN BUY! Has automatic error detection and CORRECTION of up to 11 bit burst errors. SS-50 bus, extended addressing capabilities, DMA, on board sector buffer, drivers included for 6809 FLEX or OS-9. Specify whose version of FLEX that you are using. Drivers for FLEX2 (6800) are available for an additional $100.00. Price includes host interface, controller, drive(s), and cables.

7 Megabyte single drive sys.	$1995.00	14 Megabyte dual drive sys.	$2995.00
12 Megabyte single drive sys.	$2295.00	24 Megabyte dual drive sys.	$3595.00
19 Megabyte single drive sys.	$2995.00	38 Megabyte dual drive sys.	$4695.00

(19 Megabyte drives are the largest that can be supported by FLEX)

Circuit boards, cables, software (No drives)	995.00
SS-50C DMA Bus interface board only	695.00

ELEKTRA HD-5 Cabinet for dual 5 1/4" floppy drives with power supply, fine cord, fuse power switch, and power cables to drives 150.00

ELEKTRA HD-8W As above but with EMI filter, fan, and heavy duty power supply. Powers 1 floppy and 1 Winchester. 199.00

5" ribbon cable for dual outboard 5-1/4" disk drives	40.00
2" ribbon cable for dual inboard 5-1/4" disk drives	35.00
Custom cables available	Phone

ELEKTRA HD-8 Dual 8" drive cabinet, EMI filter, fan with filter, power supply and power supply cables. 350.00

8" ribbon cable for dual 8" disk drives	45.00
ELEKTRA 30 PIN PROTOTYPING BOARD	20.00
ELEKTRA 10 PIN PROTOTYPING BOARD	40.00
GOLD 10 PIN CONNECTORS (Specify male with square pins or female)	1.50
TIN 10 PIN CONNECTORS (Specify male with square pins or female)	.50

ELEKTRA is a trademark of AAA Chicago Computer Center.
FLEX and UniFLEX are trademarks of Technical Systems Consultants, Inc.
HELIX is a trademark of Hazelwood Computer Systems.
OS-9 and **BASIC09** are trademarks of Microware Systems Corp.

AAA CHICAGO COMPUTER CENTER (312) 459-0450
120 CHESTNUT LANE ● WHEELING, IL 60090
Technical consultation available 4 PM to 6 PM most weekdays. Closed evenings and weekends.

TERMS Minimum order $20.00. Shipping and handling estimates within the Continental U.S., add 3% (MINIMUM $2.50). Illinois residents add 6% sales tax. We will refund your overestimated shipping and handling charges. Foreign shipping and handling, add 10% (MINIMUM $10.00). Foreign orders must be prepaid in U.S. dollars. Checks must be drawn on a U.S. bank. Heavy foreign items will be shipped air freight collect. Please phone between 4 PM and 6 PM weekdays if questions arise regarding shipping fees. Master Charge, Visa, and American Express honored.

Our policy: We are not staffed to answer technical inquiries through the mail. Please phone for technical help during the hours indicated above. The fast changing of our inventory and prices makes it uneconomical to publish a catalog. Our ads are intended to serve that purpose. Prices and inventory are subject to change without advance notice.

ELEKTRA ™ SOFTWARE (All of our software is copyrighted and all rights are reserved. Source is either supplied or optionally available at extra cost so that the purchaser can modify our programs for his own use. Licensing, however, is required for commercial resale.)

SUPER MODEM PROGRAM Single character commands. No interrupts required. Transmit manually or transmit disk files (text) of any length to distant computer. Receive and save disk files (text) on local disk system. X-on/X-off supported. Tested for full duplex at speeds up to 9600 baud. Half duplex option. Echo option. Replaces CR with CR/LF (user option). Slow disk file transmit option.

Please specify 6800 or 6809, SSB or FLEX™, 5" or 8".
Instruction Manual and disk with both source and object code $75.00

STANDARD MODEM PROGRAM Same as Super Modem Program above but without ECHO option, CR/LF for CR option, slow disk file transmit option, nor X-on/X-off option. specify 6800 or 6809.
Manual with instructions, source listing, and flow chart 45.00

OS-9 Configurable Modem Program (Sorry, source is not available) 100.00

ORDER — WRITE UP COMPUTER PROGRAM Screen oriented write up form with cursor editing, disk save and load, printer command using easily available universal print-out forms. Phone for more details. Available for 6809 FLEX. $100.00

ALL IN ONE
Editor — Text Processor — Mailing Labels — Mailing Lists — Multiple Form Letters
Use any CRT terminal and printer — Best Package For The Money Anywhere!
Specify 6800 or 6809, SSB or FLEX™, 5" or 8" 75.00
Add $35.00 for printed source listing; add $100 for source on disk.
All-In-One, Write'n spell, and Spell'n Fix package 250.00

Software by Technical Systems Consultants, Inc.	FLEX™ Source (List)	FLEX™ Source (Disk)	FLEX™ Man. Only	FLEX™ Object w/Man.	UniFLEX™ Add. Man.	UniFLEX™ Man. Only	UniFLEX™ Object w/Man.
Gen. DOS w/Edit & ASMB	—	—	—	250	—	—	—
SWTPC DOS w/Edit & ASMB	—	—	25	150	40	100	550
Advanced Programmers Guide	—	—	25	—	—	—	—
Editor	100	250	25	50	—	—	—
Assembler	150	250	25	50	—	—	—
Debug	175	250	25	75	—	—	—
Extended Basic	—	—	25	100	20	50	200
Basic Precompiler	—	—	25	50	10	25	150
Sort/Merge	—	—	25	75	20	35	150
Utilities	—	Inc.	25	75	10	25	150
Diagnostics	—	—	25	75	—	—	—
Text Processor	150	250	25	75	20	35	150
68000 X-ASMB on 6809	—	—	25	250	20	35	300
Pascal	—	—	50	200	25	50	300
Rel.ASMB/Linking Loader	—	—	25	150	20	35	175
6800 X-ASMB on 6809	—	—	—	100	—	—	—
Cobol	—	—	—	—	30	75	750
Fortran 77	—	—	—	—	35	65	450

Software by Microware Systems Corp. (Suggested List Prices, varies/mfg)	Run-Time Package	Source	Manual Only	Object w/Man.
OS-9™ Level 1 w/Edit, Asm, Debug		400.00	40.00	250.00
OS-9™ Level 2 w/Edit, Asm, Debug		400.00	40.00	500.00
OS-9™ Edit, Asm, Debug Pkg			25.00	125.00
Device Driver for Disk Controller (Specify Model)		100.00		
Device Driver for ACIA and PIA		50.00		
Clock Driver for 6840 and 58167 clock chips		35.00		
Entertainment Pack I, or File Handler Toolbox, or NineCom			10.00	85.00
Print Spooler (Level 2 only)			15.00	150.00
Virtual Disk Driver (Level 2 only)			10.00	85.00
RMA Relocatable Macro Assembler			20.00	200.00
RMA/68000 Cross Assembler				400.00
BASIC09™ W/Run-Time	50.00	N/A	25.00	200.00
BASIC09 Tour Guide Book			18.95	
"C" Compiler			25.00	250.00
C Programming Language (Kernighan & Ritchie)			19.95	
CIS Cobol Compiler w/Forms 2 Prog. Gen.	50.00	N/A	40.00	400.00
Pascal Compiler	50.00	N/A	25.00	250.00
Sage Application Generator	300.00	N/A	25.00	995.00
Microware yearly support service (All products)				150.00
Edition Update w/o manuals 25.00		Version Update w/manuals		75.00

Special Software

STAR-DOS Level 1 Regular Price $75.00		Introductory Price	$50.00
2K 6809 MICROBUG	40.00	4K 6809 HUMBUG	75.00
2K 6800 HUMBUG	40.00	4K 6800 HUMBUG	75.00

Other HUMBUG versions including video versions are available.

Spell'n Fix by Peter Stark 178.58	Write'n Spell by Peter Stark 75.11
All-In-One, Spell'n Fix, and Write'n Spell package	250.00
SUPER SLEUTH Disassembler System ($101.00 for OS-9 version)	99.00

SD/DD DISK DRIVES	1 head Tandon	2 heads Tandon	2 heads CDC	1 head MPI	2 heads MPI
30 day guarantee					
5-1/4", 40 tracks	225.00	300.00	300.00	250.00	325.00
5-1/4", 80 tracks	300.00	375.00	375.00	325.00	400.00
MPI or CDC Service Manual (Specify 40 or 80 track)					25.00
8", 77 tracks DS/DD	Qume DT-8 $550.00			Remex (Special)	350.00

SPECIALS

* U.S. Robotics 300/1200 baud auto dial/auto answer modem			499.00
* Same as above but without self test and diagnostics			399.00
* U.S. Robotics 1200 baud direct connect auto answer modem			299.00
* SSB BFD Floppy Disk Controller (Version 3) Run FLEX or SSB DOS			100.00
* SWTPC 4K Memory $15.00	MP-Mb (4K bareboard)		9.95
* High speed tape reader 50.00	300 Baud acoustic modem		129.00
* TI 810 Printer w/lower case and full vertical forms control			1200.00
* Microtime II Calendar and Clock Board	Assembled 60.00	Bare	30.00

GIMIX CLEARANCE SALE

	LIST	OUR PRICE		LIST	OUR PRICE
#5 6809 Plus CPU Bd.	578.05	475.00	6800 CPU board	224.03	100.00
Cable (Ser or Par I/O)	24.95	20.00	8 Port Serial I/O Bd.	318.46	250.00
Double disk reg. card	68.22	50.00	#28 control w/GMX Flex	328.28	270.00
80 X 24 Video Boards	398.76	250.00	Single prt ser, 1 cable	113.36	90.00
64 X 16 Video Boards	198.71	100.00	Dual prt par, 2 cables	138.32	110.00
16K Mem.Bds. w/cntrl reg.		145.00	4K PPD PROM Bd. and burner		100.00
93L422 RAM chips (2 needed for GIMIX DAT)				each	20.00

SWTPC

MP-S2 Dual Port Serial	120.00	MP-L2 Dual Port Parallel	120.00
MP-N Calculator Board (kit)	54.95	S-32 RAM not included	124.50
MP-R 2716 Eprom Programmer	114.50	MP-09 2MHz 6809 CPU Board	295.00

Smoke Signal Broadcasting

DCB-4A Double Density Controller Board for 5" and 8" with DOS	549.00
SCB-69 6809 CPU Board	399.00

WARNING AAA Chicago Computer Center does not provide repair or diagnostic service for customer assembled kits. AAA Chicago Computer Center does warranty and maintain service for our assembled boards. The customer should carefully take into consideration the small differential separating our kit and assembled prices when making his choice of purchase.

We have introduced our line of computer equipment with the purpose of offering the highest quality of components possible at affordable prices. These products are intended for OEM applications where it is the responsibility of the purchaser to integrate these components with suitable memory, disk controllers, drives, and software along with I/O terminals to form working computer systems.

buy very cheaply such a drive; as you do not count in dollars the time you spend on repair, you may get some interesting insights into the inner workings of your machine, and some really good vibrations when your stone-dead drive is resuscitated !

- - -

(**) Editor's Note: I believe! - A few years back I purchased, at a computer show from a fellow cruising the isles, a pair of these drives, brand new, sealed in the box, with data sheets and quality control(?) stickers attached. (At a real good price I must admit)

Upon arriving back at the office the following Monday I licked my chops, as I unpacked the new MPI drives, could hardly wait to double my disk storage. Which was strained as 68 Micro Journal was growing and my 'work at home' system was supplied only with single-sided 5" drives, later expanded to DS/DD 5&8" drives and a 20 MByte winchester. This was in the days when double sided, double density drives were not only expensive but short due to demand. Finding these two was a real bonus from the show.

After about an hour of figuring out how to program the drives for side select, etc., and getting them installed in the power supply cabinet, I open the door to drive #0, inserted the 'System' disk, typed 'U' for 5" disk and waited, and waited and waited some more. The motor buzzed and hummed, the LED function light lit up as befits a proper disk drive but nothing else. Somewhat downcast I reached up and opened the drive door - maybe the diskette wasn't seated just right, when - ZAP, BANG, FLOP, ZOOOOOMM >, out of the door opening flew a steel rod, a couple of machined parts and assorted nuts, bolts, screws and lockwashers! Talk about surprise!

Spent nearly an hour sifting and sorting around down on the floor searching for the 'flying' drive parts. Got into all sorts of problems as I finally came up with about twice as many nuts, screws and washers, as there were empty holes in the drives. Happily the other drive had not reacted so violently, so I 'carefully' removed it also from the cabinet. This was to be my guide to where, whatever, was to be replaced. It appeared to be intact and not prone to self-destruction so with a large sigh of relief I commenced to reassemble the sick drive - but wait, none of the screws will fit in the holes properly, and what was even more upsetting was that on inspection there were in some places no holes, drilled or cast, where they were on the good drive. And to make matters worse most of the holes that did exist had no threads!

Mind you this drive was factory sealed and had 'qc' stickers pasted all over.

Spent the next couple hours drilling holes and cutting threads to proper size. Reassembled the drive per the good one, did an eyeball alignment and reinstalled the drives in the cabinet, punched 'U', watched the LED light up, listened to the motor whirr and purr - and behold the screen displayed the proper FLEX™ banner. IT WORKE !!

Used these drives without any faltering on their part for nearly two years. Then one day it became apparent that the original 'good' drive had slipped sync or was out of head alignment. Upon removing it from the cabinet, after a two year or so of faultless operation, I observed a shiny small brass piece dangling on the end of a fine thread of wire - yep - was the pole piece of one of the heads. Too much this time - back to the factory. Actually both drives as I discovered that the other, original bad, drive was also loosing its pole piece. Seems the white or light plastic the pole pieces had been imbedded in had turned to granulated small particles, much as sugar or salt. Thus allowing the pole pieces to literally fall out, and dangle on the pole coil wire.

After about a month of waiting and having complied with their request that I call first and secure a service authorization number, I started calling every day or so, but never could get a straight answer. Finally I called on a Saturday afternoon, and got in contact with a young man in the 'service department', who informed me that he was a high school student and helped out occasionally in the service department. According to what he told me there was only one full time service man and he was 'snowed under'. After giving him my authorization number he requested that I call back in a couple of hours, "As there is a large pile of drives behind the door and I will have to go through them all to find yours." I called back a few hours later but no one answered the phone. I was beginning to wonder.

The following Monday I called and gave the nice young lady who answered the phone a quick rundown on the 'fun' I had been having with my drives. Also told her that I was filing a complaint with the FTC and any other threat I thought might have leverage. IT worked as she called me back within the hour. She informed me that my drives had been misplaced, but would be rushed to the front of the line, repaired, and returned post hasto...

Well to make a rather boring story short I did receive a parcel from them a few weeks later. Inside were two drives, not my old ones, but new current models. Also I received a letter from some official explaining that my instance was an isolated one. Seems that some employee had pilfered some production line drives and sold them to some one else, who sold them to an unsuspecting(?) individual who hawked them off at a computer show. I bought a couple. I knew others who also bought a couple and checking with them, things were running smooth - no complaints. For that reason I never did say anything about it in 68 Micro Journal. Maybe it was an isolated case but lighting sure struck twice over at my place.

- - -

BIT BUCKET

SPECTRUM Electronics

March 16, 1984

Computer Publishing Center
68' Micro Journal
5900 Cassandra Smith
P.O.Box 849
Mixson. TN
USA , 37343

Dear sirs:

I have received a sample copy of your January '84 issue and find the advertisements most enlightening. Prior to this I felt that my SWTP 6800/6809 system was old, and good for a boat anchor!

The review of CCSM Standard MUMPS by Peter Dibble compels me to brush away the cob webs and try to make use of my hardware for business applications. Where can I get a version of MUMPS that will run on my Percom LFD 400, either with MINIdos 1.4 (6800), or MPX-9 (6809)?

Will their be regular column related to MUMPS? If I can't obtain MUMPS then I will most likely opt for FLEX and subscribe to your informative magazine.

Your speedy reply will be most appreciated.

Sincerely,

E.Vander Ryd / mgr

EVR/sf

14 Knightswood Crescent, Brantford, Ontario, Canada, N3R 7E6

Glad to see that you have come to realize that there is a lot of life still left in the Standard S50 Bus machine you own.

You will find that there is available for your SWTPC computer practically any applications program you need.

MUMPS has caught on with quite a few, but we will not be devoting a regular column to MUMPS. We will be looking for articles on MUMPS for future publication however.

As to PERCOM support, for any language or operation, I am afraid you will find little if any remaining. PERCOM left the Standard S50 Bus some time back, and due to their small following no one stepped in to fill the void.

As to your use of FLEX™, well there is still support for FLEX, and FLEX has a wealth of applications type programs available from varied sources. See 68 MICRO JOURNAL. FLEX has gained such a wide base of support that it will survive as a viable disk system, with or without originator support. FLEX is not as well supported by TSC, in my opinion, as is some of the other systems, i.e Microwares's OS9, which is gaining in popularity as more and more users switch or new users come along. However, you would do well with FLEX, it is a proven, very popular disk system, and has a larger user base on Standard S50 Bus computers, than practically any other. On non Standard S50 Bus computers OS9 se s to be the most popular. However, in the next year or so things might change somewhat, so you really have only good choices remaining. Good luck!

OMW

- - -

MOTOROLA INC.
MOS Integrated Circuits Group
3501 ED BLUESTEIN BLVD
AUSTIN, TEXAS 78721
For further information contact

Press Information

Editorial Contact: Susan Dunn
512 928-6804
Reader Contact: Wade Wyatt
512 982-6860

Motorola Announces the
Raster Memory System (RMS)
Graphics Chip Set

Austin, Texas, January 26, 1984... Motorola Microprocessor Division announces a new concept for graphics/text generation and display. A 2-chip set, the Raster Memory System (RMS) is composed of the MC68487 Raster Memory Controller (RMC) fabricated in HCMOS, and the MC68486 Raster Memory Interface (RMI), implemented in Motorola Oxide-Isolated Self-Aligned Implanted Circuit (MOSAIC) bipolar technology. This low-cost, high-resolution chip set contains a myriad of features to provide the control for high-quality CRT graphics. Its capabilities are so pervasive that the graphics quality is primarily dependent on the power of the MPU, the amount of memory available, and the sophistication of the software, not by any limitations within the RMS system itself.

The RMS has been designed for personal computer, videotex, games, workstation, computer animated graphics and word processing markets.

"We feel that the RMS, with its many capabilities and low price tag, will revolutionize the market and set a new industry standard for intermediate graphics applications," states Steve Marsh, Marketing Manager for Motorola's Microcomputer Group.

The RMS operates with most popular CRT's. It will drive monitors with a 50 Hz field refresh rate used in Europe, or with a 60 Hz rate popular in the U.S. There is a choice of resolutions up to 640 pixels per line, and up to 500 lines per screen. For text reproduction, screen resolutions of either 32, 40, 64, or 80 characters per line are available in full color. For graphics interpretation, choices of 256, 320, 512, or 640 pixels per line provide the needed definition to fit a variety of applications. These choices are software selectable from internal horizontal and vertical resolution registers.

The RMS chip set has been designed to operate with Motorola's M6809 and M68000 class of microprocessors. Flexibility is inherent in the RMS due to the variety of operating modes available. The RMS is backward compatible with Motorola's popular MC6847/MC6883 existing graphics solution. However, the use of high-performance CMOS technology in the RMC permits many new enhancements. The Raster Memory System can operate in either a bit-plane mode, where each pixel of display is governed by a unique location of RAM, or in one of six character-oriented modes called list modes. If one of the list modes is selected, the user can choose from 96 alphanumeric characters and 64 MOSAIC characters stored in the RMC's internal ROM. In addition to these, up to 32K dynamically redefinable characters can be supported. All of these character types can have attributes applied to them, such as color, underline, flash, invert, double high, and/or double wide.

The RMS chip set can simultaneously display up to 32 colors from its palette of 4096 colors. Color data selected is stored in a Color Mapping RAM which is addressed at the pixel rate as the video picture is generated. This extremely color selective system permits the programmer to use adjacent pixels which are only slightly different in shade, as in flesh tones, or completely different in color, as in the reproduction of company logos.

The RMS has two screen definitions; the virtual screen and the actual (display) screen. The virtual screen is usually much larger so that the actual screen can be positioned anywhere within the virtual screen. The effect is to be able to smoothly scroll in any direction without continuously changing memory. Up to 1 megabyte of DRAM memory can be accommodated. It is possible to sync the RMS to an external source to allow overlay of RMS data onto an existing VCR or broadcast TV picture. Fixed are redefinable characters with unique attributes. Such object attributes as color, priority, shading and collision detection make them especially useful in game applications.

The RMI is a bipolar chip that provides high-speed interfaces between the external DRAM, MPU, RMC. The RMI provides red, green, blue for the MPU, DRAM, and RMC and synchronization signals to the monitor, and provides interface signals.

The price of the RMC and RMI is expected to be in the $20 range in high volume within a year from full production. Samples will be available in the 4th quarter of 1984 and production is slated for the 1st quarter of 1985.

Contact Literature Distribution Center in Phoenix, 616 W. 24th St., Tempe, AZ 85282 for brochures, or your local Motorola Sales Office for further information.

User-entered Filenames Under SWTPC BASIC 3.5

By Geoffrey A. Gass
5240 S. W. Dosch Rd.,
Portland, Oregon 97201

Those of us who thought we'd work up some clever program in SWTPC BASIC 3.5 to use operator-supplied FLEX filenames, got a quick come-uppance. This 6800/FLEX disk BASIC by Bob Uiterwyk does not support use of a string variable in an OPEN statement. So normally, there's no way the operator of an application program can enter a name for a new data file except by escaping to the BASIC editor and changing the program itself.

However, if you're willing to live a little dangerously and in violation of some cardinal rules of programming, here's a process by which operator input can provide the filename for a data file generated during operation of a program.

We can write the program using a dummy filename, and then use PEEK and POKE statements to change that program line as required. This is called "self-modifying code", and in some shops will get you thirty lashes and a dock in pay ... so be warned.

First of all, we need a routine to find the OPEN statement. This allows the statement to be located anywhere, and allows program changes that would relocate it.

The statement is not found by looking for the letters O-P-E-N. The first verb on each line is "tokenized" in Uiterwyk's BASIC interpreters, and the token is the two-byte address in the command table where the verb and the address of its implementing routine are stored. In my version of BASIC 3.5, the token for OPEN is hexadecimal 0339, and the two bytes we are looking for are decimal 03 and 57.

Unfortunately, these tokens are subject to change when the command table is modified. To find

the token in your version, enter the statement:

```
1 OPEN #1, DDD
```

Now, examine hex location 002E to find out where line 1 is, and examine the contents of that line, which will be:

00	01	tt	tt	LL	23	31	2C	20	44	44	44
(Line #)	(Token)	(*)	#	1	,	(SP)	D	D	D		

* = Line length

The token is displayed by the monitor as two hexadecimal bytes. Convert these to decimal ... or let BASIC do it for you! In direct mode, enter:

```
A=256*PEEK(46)+PEEK(47): PRINT PEEK(A+2);PEEK(A+3)
```

So now, up in the initialization section, we can put in lines 255-295 (see listing).

This routine sets up variable A8 to point to the start of the first position for a possible filename. If the line containing the OPEN statement is well into the program, the intialization time will take 20 seconds to several minutes. To minimize the delay, make the OPEN statement a quickie subroutine near the start of the program:

```
100 GOTO 120
110 OPEN #1, FILENAME.EXT : RETURN
120 ...
```

It is critical that the dummy filename be entered just as shown here, to hold the space for the one which will be supplied by the operator.

The second part of the chore is to build a filename-checker routine, to intercept any illegal-format names. Otherwise, BASIC may butt in with a stupid "ERROR #X ON LINE Y # READY", which the operator may not be prepared to cope with.

So, we add the subroutine consisting of lines 1500-1870 (see listing). This code does not allow use of the underscore character, and requires a 3-letter extension to be explicit (no default extension). If you desire, some added code can make this routine less restrictive. As it stands, it

does guarantee that an acceptable filename is presented to FLEX.

Finally, we add the subroutine which enters the filename into the program code. This is contained in lines 1900-1990 of the listing. The routine always enters 12 characters, with leading spaces if the filename N$ is short. Just before calling the OPEN statement, it resets A8 so it can be used again.

The routines herewith get most of the job done. An additional checking routine might be worthwhile to forestall an operator trying to open a new file before closing the old one. A perfectly foolproof system would also look to see if the disk drives were ready before allowing an OPEN operation -- but I'll leave that for another time.

```
0010 REM *        USERNAME          *
0020 REM *   Routines to enter      *
0030 REM *   Operator-furnished     *
0040 REM *  Filenames under SWTPC   *
0050 REM *  BASIC 3.5 (6800/FLEX)   *
0060 REM
0070 REM *   By Geoffrey A. Gass    *
0080 REM * Portland, Oregon  97201 *
0090 REM

0099 REM *   OPEN statement kept up front   *
0100 GOTO 120
0110 OPEN #1, FILENAME.EXT : RETURN

0254 REM * Finding the OPEN statement *
0255 A6 = PEEK(47) + 256*PEEK(46) : REM Start of this program
0260 A9 = PEEK(43) + 256*PEEK(42) : REM End of this program
0265 A7 = A6 : REM Point to start of line
0270 IF PEEK(A7+2) <> 03 THEN 280 : REM Looking for OPEN token
0275 IF PEEK(A7+3) = 57 THEN 295 : REM Token is hex 03 9
0280 A7 = A7 + PEEK(A7+4) : REM 5th byte is distance to next line
0285 IF A7 < A9 THEN 270 : REM Not to end yet
0290 PRINT "No 'OPEN' statement."
0295 A8 = A7+9 : REM Start of filename if correctly entered

0999 REM *    START NEW DATA FILE    *
1000 GOSUB 9000 : REM Clear screen
1010 GOSUB 1500 : REM Get filename
1020 GOSUB 1900 : REM Open file
1030 REM Perform other file initialization chores
1090 RETURN

1499 REM *     Get Filename      *
1500 GOSUB 9000 : PRINT
1510 INPUT "Name of data file",N$
1520 IF LEN(N$) > 12 THEN 1800
1530 IF LEN(N$) < 1 THEN 1510
```

```
1540 IF LEFT$(N$,1) < "A" THEN 1800
1550 N=0 : P = 0 : X = 0 : E = 0
1560 FOR I = 1 TO LEN(N$)
1570 S$ = MID$(N$,I,1)
1580 IF P > 0 THEN 1630 : REM This is extension
1590 IF S$ = "-" THEN 1700
1600 IF S$ = "." THEN P = P + 1 : GOTO 1680
1610 IF S$ < "1" THEN 1670 : REM ERROR
1620 IF S$ <= "9" THEN 1700 : REM OK
1630 IF S$ < "A" THEN 1670 : REM ERROR
1640 IF S$ <= "Z" THEN 1700
1650 IF S$ < "a" THEN 1670
1660 IF S$ <= "z" THEN 1700
1670 E = E + 1 : GOTO 1720
1680 IF N < 1 THEN E = E + 1
1685 IF P > 1 THEN E = E + 1
1690 GOTO 1720
1700 IF P > 0 THEN X = X + 1
1710 IF P = 0 THEN N = N + 1
1715 IF N > 8 THEN E = E + 1
1720 NEXT I
1725 IF E > 0 THEN 1800
1730 IF X = 3 THEN RETURN
1740 IF P = 0 THEN N$ = N$ + "." : P = 1
1750 IF X <> 0 THEN 1800
1760 INPUT "Filename extension",D$ : N$ = N$ + D$
1770 FOR I = 1 TO LEN(D$) : S$ = MID$(D$,I,1)
1780 GOTO 1580
1800 PRINT "A filename must be 1 to 8 characters long."
1810 PRINT "It must start with a letter."
1820 PRINT "It may contain only letters, numbers and hyphens."
1830 PRINT "A filename extension must be 3 letters long."
1840 PRINT "Please enter a filename and extension conforming"
1850 PRINT "to FLEX requirements."
1860 INPUT "OK" , N$
1870 GOTO 1500

1899 REM Self-modifying BASIC code enters filename
1900 IF LEN(N$) = 12 THEN 1950
1910 FOR J = 12 TO LEN(N$)+1 STEP -1
1920 POKE( A8,32)
1930 A8 = A8 + 1
1940 NEXT J
1950 FOR J = 1 TO LEN(N$)
1960 POKE( A8,ASC(MID$(N$,J,1))) : A8 = A8 + 1
1970 NEXT J
1980 A8 = A7 + 9 : RETURN
1990 GOTO 110 : REM Open file

8999 REM *     Erase Routine     *
9000 PRINT CHR$(16);CHR$(22)
9010 RETURN
```

Getting out of XBASIC.

Have you ever wanted to be able to get out of XBASIC, and go back to Flex under the control of the XBASIC program? I did recently in setting up an EXEC file, and couldn't find any command to do it.

The following routine should be self-explanatory. What easier way to cut off an XBASIC program than to use a CHAIN SAM?

```
5000 REM  ************************************************************
5010 REM  *                                                          *
5020 REM  *                        S A M.                            *
5030 REM  *                                                          *
5040 REM  *    A program to return gracefully to Flex from XBASIC,   *
5050 REM  *         under the control of an XBASIC program.          *
5060 REM  *                                                          *
5070 REM  *    The program should be entered, compiled with the      *
5080 REM  *      name SAM.BAC, and included on the working disk.     *
5090 REM  *                                                          *
5100 REM  *    To exit to Flex, include the following line in        *
5110 REM  *      the XBASIC program:                                 *
5120 REM  *                                                          *
5130 REM  *    <line number> CHAIN SAM                               *
5140 REM  *                                                          *
5150 REM  *              Whitethorn Software  -  1984.               *
5160 REM  *                                                          *
5170 REM  ************************************************************
5180 REM
5190 POKE HEX("C100"),HEX("7E")      : REM  Put Flex warm start
5200 DPOKE HEX("C101"),HEX("CD03")   : REM    address at $C100.
5210 A = DPEEK(HEX("CC2B")) - 2      : REM  End of memory - 2.
5220 DPOKE A,HEX("C100")             : REM  Point to sub-routine.
5230 A = USR(0)                      : REM  Execute it.
```

Yours sincerely,

Alan M. Fowler.

Whitethorn.
3 Lemon Road,
North Balwyn. Vic. 3104.
AUSTRALIA.

Telephone +613-857-7128 (Home)

18th January, 1984.

MATERIALISM

POSSESS is a program that allows one to build up an inventory of one's personal belongings for insurance purposes, etc. The entries are made in DATA statements starting at LINE 1690 and include the name of the item, the manufacturer of the item, the serial number of the item, the quantity of the item, and the dollar value of the item. The entire file, or items made by a particular manufacturer, or the item type, or items having a value greater than an entered value can be listed either to the CRT or to a printer. Items that have a serial number and presumably have a large value should be entered as single items. The dollar value of a lot of items can be assigned as the total value or the value per item. This is left up to the user but consistency of assignment is suggested.

Jeffrey M. Craig
Apt. 912 - 3001 S. King Dr
Chicago, IL 60616


```
10 LET N=0
20 DIM I$(3);DIM J$(3);DIM M$(3);DIM N$(3)
30 PRINT "THIS PROGRAM WAS WRITTEN BY JEFFREY M. CRAIG"
40 PRINT "JANUARY 26,1984"
50 PRINT "PRESENT ADDRESS: APT.912 - 3001 S.KING DR."
60 PRINT "CHICAGO, IL 60616"
70 PRINT "INVENTORY OF PERSONAL BELONGINGS"
80 PRINT "AS OF MARCH 14, 1982"
90 PRINT "THIS PROGRAM IDENTIFIES THE OBJECT BY ITEM,"
100 PRINT "MANUFACTURER, SERIAL NUMBER, QUANTITY, AND VALUE"
110 PRINT "IF YOU WANT HARDCOPY OF THE EXECUTED PROGRAM"
120 PRINT "THEN ENTER 'Y' - IF YOU WANT A CRT DISPLAY ENTER"
130 PRINT "'N' FOR NO."
140 INPUT P$
150 IF P$<>"Y" AND P$<>"N" THEN GOTO 110
160 PRINT "DO YOU WANT TO SEARCH FOR A PARTICULAR ITEM, MANUFACTURER,"
170 PRINT "OR ITEMS OF A GREATER DOLLAR VALUE THAN THAT"
180 PRINT "REQUESTED?"
190 PRINT "(IF YOU ENTER 'N' THE PROGRAM WILL BEGIN PRINTING"
200 PRINT "THE ENTIRE LISTING OF ITEMS, MANUFACTURERS, SERIAL"
210 PRINT "NUMBERS AND VALUES FROM DATA STATEMENTS)"
220 PRINT "ENTER 'Y' FOR YES OR 'N' FOR NO."
230 INPUT A$
240 IF A$<>"Y" AND A$<>"N" THEN GOTO 160
250 IF A$="N" THEN GOTO 430
260 PRINT "DO YOU WANT TO SEARCH FOR AN ITEM?"
270 PRINT "ENTER 'Y' FOR YES, 'N' FOR NO."
280 INPUT B$
290 IF B$<>"Y" AND B$<>"N" THEN GOTO 260
300 IF B$="Y" THEN GOTO 620
310 PRINT "DO YOU WANT TO SEARCH FOR A MANUFACTURER?"
320 PRINT "ENTER 'Y' FOR YES, 'N' FOR NO."
330 INPUT C$
340 IF C$<>"Y" AND C$<>"N" THEN GOTO 310
350 IF C$="Y" THEN GOTO 930
360 PRINT "DO YOU WANT TO SEARCH FOR ITEMS OF GREATER VALUE"
370 PRINT "THAN AN ENTERED VALUE?"
380 PRINT "ENTER 'Y' FOR YES, 'N' FOR NO."
390 INPUT D$
400 IF D$<>"Y" AND D$<>"N" THEN GOTO 360
410 IF D$="Y" THEN GOTO 1240
420 GOTO 1420
430 IF P$="Y" THEN OPEN "O,PRINT" AS 0
440 N=-99
450 READ I$,M$,S$,Q,V
460 IF I$="END" THEN GOTO 1480
470 IF P$<>"Y" THEN GOTO 500
480 GOSUB 1460
490 GOTO 510
500 GOSUB 1550
510 PRINT "ENTER 'CONT' TO KEEP LISTING ITEMS"
520 STOP
530 GOTO 450
540 PRINT "ALL ITEMS IN THE LISTING HAVE BEEN EXAMINED"
550 PRINT
560 PRINT "DO YOU WISH TO EXIT PROGRAM?"
570 PRINT
580 PRINT "ENTER 'Y' FOR YES, 'N' FOR NO."
```

```
590 INPUT E$
600 IF E$<>"Y" AND E$<>"N" THEN GOTO 540
610 IF E$="Y" THEN GOTO 1400
620 RESTORE
630 PRINT "ENTER NAME OF ITEM TO BE SEARCHED"
640 INPUT J$
650 IF P$="Y" THEN OPEN "0,PRINT" AS 0
660 FOR I=1600 TO 9999 STEP 50
670 READ I$,M$,S$,Q,V
680 IF I$="END" THEN GOTO 540
690  I$(I)=LEFT$(I$,1)
700  J$(I)=LEFT$(J$,1)
710 IF I$(I)<>J$(I) THEN GOTO 840
720  I$(2)=LEFT$(I$,2)
730  J$(2)=LEFT$(J$,2)
740 IF I$(2)<>J$(2) THEN GOTO 840
750  I$(3)=LEFT$(I$,3)
760  J$(3)=LEFT$(J$,3)
770 IF I$(3)<>J$(3) THEN GOTO 840
780 IF P$<>"Y" THEN GOTO 810
790 GOSUB 1460
800 GOTO 820
810 GOSUB 1550
820 PRINT "ENTER 'CONT' TO CONTINUE SEARCH"
830 STOP
840 NEXT I
850 PRINT "ALL MANUFACTURERS HAVE BEEN EXAMINED"
860 PRINT
870 PRINT "DO YOU WISH TO EXIT THIS PROGRAM"
880 PRINT
890 PRINT "ENTER 'Y' FOR YES, 'N' FOR NO."
900 INPUT F$
910 IF F$<>"Y" AND F$<>"N" THEN GOTO 870
920 IF F$="Y" THEN GOTO 1400
930 RESTORE
940 PRINT "ENTER NAME OF MANUFACTURER TO BE SEARCHED"
950 INPUT N$
960 IF P$="Y" THEN OPEN "0,PRINT" AS 0
970 FOR I=1600 TO 9999 STEP 50
980 READ I$,M$,S$,Q,V
990 IF I$="END" THEN GOTO 850
1000  M$(1)=LEFT$(M$,1)
1010  N$(1)=LEFT$(N$,1)
1020 IF M$(1)<>N$(1) THEN GOTO 1150
1030  M$(2)=LEFT$(M$,2)
1040  N$(2)=LEFT$(N$,2)
1050 IF M$(2)<>N$(2) THEN GOTO 1150
1060  M$(3)=LEFT$(M$,3)
1070  N$(3)=LEFT$(N$,3)
1080 IF M$(3)<>N$(3) THEN GOTO 1150
1090 IF P$<>"Y" THEN GOTO 1120
1100 GOSUB 1460
1110 GOTO 1130
1120 GOSUB 1550
1130 PRINT "ENTER 'CONT' TO CONTINUE SEARCH"
1140 STOP
1150 NEXT I
1160 PRINT "THE ENTIRE LISTING OF ITEMS HAS BEEN EXAMINED"
1170 PRINT
1180 PRINT "DO YOU WISH TO EXIT PROGRAM"
1190 PRINT
1200 PRINT "ENTER 'Y' FOR YES, 'N' FOR NO."
1210 INPUT G$
1220 IF G$<>"Y" AND G$<>"N" THEN GOTO 1160
1230 IF G$="Y" THEN GOTO 1400
1240 RESTORE
1250 PRINT "ENTER SELECTED VALUE IN DOLLAR AMOUNTS"
1260 PRINT "(DO NOT ENTER THE DOLLAR SIGN)"
1270 INPUT U
1280 IF P$="Y" THEN OPEN "0,PRINT" AS 0
1290 FOR I=1600 TO 9999 STEP 50
1300 READ I$,M$,S$,Q,V
1310 IF I$="END" THEN GOTO 1160
1320 IF V<U THEN GOTO 1390
1330 IF P$<>"Y" THEN GOTO 1360
1340 GOSUB 1460
1350 GOTO 1370
1360 GOSUB 1550
1370 PRINT "ENTER 'CONT' TO CONTINUE SEARCH"
1380 STOP
1390 NEXT I
1400 IF N=0 THEN GOTO 1420
1410 PRINT "ALL ITEMS IN THE LISTING HAVE BEEN EXAMINED."
1420 IF P$="Y" THEN CLOSE 0
1430 PRINT "YOU HAVE EXITED PROGRAM."
1440 PRINT "TYPE 'RUN' TO RE-ENTER PROGRAM"
1450 END
1460 PRINT #0,"                    "
1470 IF N=-99 THEN GOTO 1490
1480 PRINT #0,"LINES ";I;" THROUGH ";I+49
1490 PRINT #0,"THE ITEM IS: ";I$
1500 PRINT #0,"THE MANUFACTURER IS: ";M$
1510 PRINT #0,"THE SERIAL NUMBER IS: ";S$
1520 PRINT #0,"THE QUANTITY OF THE ITEM IS: ";Q
1530 PRINT #0,"THE VALUE OF THE ITEM IS: $ ";V
1540 RETURN
1550 IF N=-99 THEN GOTO 1570
1560 PRINT "LINES ";I;" THROUGH ";I+49
1570 PRINT "THE ITEM IS: ";I$
1580 PRINT "THE MANUFACTURER IS: ";M$
1590 PRINT "THE SERIAL NUMBER IS: ";S$
1600 PRINT "THE QUANTITY OF THE ITEM IS: ";Q
1610 PRINT "THE VALUE OF THE ITEM IS: $ ";V
1620 PRINT
1630 RETURN
1640 REM DATA NAME OF ITEM
1650 REM DATA MANUFACTURER
1660 REM DATA SERIAL NUMBER
1670 REM DATA QUANTITY
1680 REM DATA DOLLAR VALUE EACH
1690 DATA BOXCAR
1700 DATA GREENVILLE CAR COMPANY
1710 DATA 123456
1720 DATA 1
1730 DATA 10000
1740 DATA LOCOMOTIVE
1750 DATA LIMA
1760 DATA 2000
1770 DATA 1
1780 DATA 100000
1790 DATA CABOOSE
1800 DATA U.S. FOUNDRY
1810 DATA 456321
1820 DATA 1
1830 DATA 25000
1840 DATA END
1850 DATA END
1860 DATA 0
1870 DATA 0
1880 DATA 0
```

cost single-chip microcomputers, the MC6805K2. This unique 40-pin microcomputer is ideal for use in low-end and mid-range 8-bit systems designed for consumer, industrial and automotive applications. It contains a CPU, on-chip oscillator, 96 bytes of RAM, 2K bytes of user ROM, and 128 bytes of securable user EEPROM that has byte and bulk programming and erasure capabilities. The MC6805K2 is designed for 5-V only operation.

Its capability includes 32 bidirectional I/O lines which are divided into 4 8-bit ports. Each of the 32 I/O lines can be individually programmed as an input or output. All I/O port lines are CMOS and TTL input compatible and TTL output compatible. Additionally, Port A may be configured for CMOS output compatibility. Port B may also be used for interfacing with LEDs or darlingtons because it is capable of sinking and sourcing large currents. Port C is electrically similar to Port A, however, it may not be configured for CMOS output compatibility. All (8) Port D terminals may be simultaneously used as open drain outputs.

Two important features of the MC6805K2 are its flexible 4-channel, 8-bit analog-to-digital (A/D) converter and its Serial Peripheral Interface System (SPI). The Serial Peripheral Interface in this part is capable of functioning in the following 7 operating modes:

a) 1 wire - autoclocked (e.g. NRZ)
b) 2 wire - half duplex *
c) 2 wire - half duplex with clock arbitration
d) 3 wire - half duplex with slave select/busy line
e) 3 wire - full duplex *
f) 3 wire - full duplex with clock arbitration
g) 4 wire - full duplex with slave select/busy line

* Half/full duplex terminology used here is not the same as that for standard ACIA, but corresponds to direction of instantaneous information flow.

Additionally, the MC6805K2 provides an internal bootstrap loader program which allows loading of the RAM from an external source. The K2 also has self-checking capability, which allows this device to check its RAM, (internal) ROM, timer, and I/O ports for functionality.

The MC6805K2 has two programmable 8-bit timers and one 7-bit software programmable prescaler. They can be used in various configurations:

- 2 independent 8-bit timers one of which is driven by a 7-bit prescaler.
- An 8-bit timer driven by another 8-bit timer functioning as an 8-bit prescaler clocked by a 7-bit prescaler.
- A 16-bit timer driven by a 7-bit prescaler.
- A pulse width modulation digital to analog converter *.

(*) requires an external low-pass filter

The basic structure of both timers is similar. Each timer has one modulus latch and one capture latch. Each timer is an 8-bit programmable down counter which can be loaded under program control. The timers have a modulus latch which memorizes the last write to the timer data register. The contents of the modulus latch are loaded to the timer in various ways as defined by the timer control register. Each timer has a capture latch which, when enabled, memorizes the contents of the timer as a result of an external event. When disabled, it is transparent and its contents are equal to the current value of the timer. A timer overflow pulse is generated at the transition of timer to "00" state. Timer read can occur at any time "on the fly" without disturbing the countdown mechanism of the timer. Timer read does not disturb the contents of the modulus latch. At resets each timer and their modulus latch contents are set to $FF.

Samples will be available in the second quarter of 1984, and priced at $14.95 each in quantities of 1000.

Contact your Motorola Sales Office or local distributor for further information.

Dear Mr. Williams:

I am one of those persons who still haven't reached the limit of capability of the 6800 micro-processor. I'm not quite ready to up-grade yet.

The problem is, of course, that software gets scarcer and scarcer! Can you provide the address for any of the original 6800 users groups which might have programs available for free or for sale.

Sincerely yours,

Jerome Korman

415 Barberry Ave.
Kalamazoo, MI 49002
February 27, 1984

A. G. WOOD,
P. O. BOX C294,
CLARENCE ST.,
NSW 2000,
AUSTRALIA.
16/2/1984.

'68' Micro Journal,
5900 Cassandra Smith,
Hixson, TN 37343.

Dear Sirs,

I have just received your Feb'84 issue. and I see that there is still interest in prime number programs.

I would like to point out some errors which appeared in the tables reported in BYTE in Jan'83, relating to 6809 and 68000.

In the BYTE article I was mentioned as the source of the results for 68000 and IBM 5033 assembly language programs. Unfortunately, both of the results I contributed were incorrectly reported. Firstly, my results, like those of Richard Maurer (see Jun'83 BYTE) have not been "adjusted for 10 iterations". Secondly the figure given as being from an 8Mhz 68000, was actually from a 1Mhz 6809!

The "factor of 10" error appears to have been made for some of the other results in their tables. for example, two results were given for Fortran on the 5033, the one figure is about 10 times the other. I believe that many of these errors are due to the fact that in the original "Eratosthenes" article (Sep'80 BYTE), it was not made 100% clear that the figures in the tables were for 10 iterations.

I wrote to both BYTE and Jim Gilbreath, pointing out these errors, but they did not bother to publish a correction.

TALBOT MICROSYSTEMS

6800 SOFTWARE
1927 CURTIS AVENUE
REDONDO BEACH, CA 90278
(213) 376-9941

March 3, 1984

NEW PRODUCT ANNOUNCEMENT

COMPILER, INTERPRETER, AND ASSEMBLER
FOR EPSON HX-20 NOTEBOOK COMPUTER

Announcing a rom based language system for fast interactive code development for data collection, analysis, communications, and computation for the EPSON HX-20 notebook computer.

System is 10-15 times faster than BASIC!

Contains full FIG-FORTH Standard interpreter and compiler, a complete 6800/6801/6301 assembler, and many extensions for HX-20 I/O devices. Supplied in two CMOS 8K EPROMS plus a 64 page Reference Manual. May be installed in place of BASIC in the HX-20 case (leaving you about 12K of RAM working space) or in the HX-20 Expansion Unit (about 28K of RAM working space).

It provides full, detailed access to the HX-20's ROM system routines. There are families of words for control of the LCD screen graphics as well as alphanumerics, internal printer, microcassette tape, communications port, function keys, and sound generator. The assembler is designed for use to produce CODE words for even greater speed. There is a full string handling package.

For dedicated applications, a Target Compiler development system for the HX-20 or any other 6801/6301 hardware is available for 280 CP/M-80 or 68000 CP/M-68K 8" disk systems.

tFORTH/HX20 EPROM set and Manual:	$190.00
tFORTH/HX20 Manual only:	$20.00

Please add $3.00 for shipping, $12.00 foreign. VISA and MC ok.

Talbot Microsystems, 1927 Curtis Ave., Redondo Beach, CA 90278
(213) 376-9941.

March 1, 1984
946 Evans Rd
Nashville, TN 37206

Mr Larry Williams
'68' Micro Journal
5900 Cassandra Smith Rd
Hixson, TN 37343

Dear Mr Williams:

Here is something that I think will fit in the "Bit Bucket." I needed to sort strings in BASIC without regard to cases! A should equal a, and Z should follow z. TSC Extended BASIC for FLex does have string compare operators, but they compare the ASCII order of characters. Because TSC BASIC does not have a elementary function, I needed to write my own version.

My first idea was to do it all in BASIC. The code looked a little like this:

```
1000 LET T1$ = "" : LET T2$ = "" : REM TEMPORARY STRINGS
1010 LET L1% = LEN(A$(I%))
1020 IF L1% = 0 THEN 1080
1030 FOR K% = 1 TO L1%
1040 LET A9% = ASC(MID$(A$(I%),K%,1))
1050 IF (A9% > 96) AND (A9% < 123) THEN LET A9% = A9% - 32
1060 LET T1$ = T1$ + CHR$(A9%)
1070 NEXT K%
1080 LET L1% = LEN(A$(J%))
1090 IF L1% = 0 THEN 1150
1100 FOR K% = 1 TO L1%
1110 LET A9% = ASC(MID$(A$(J%),K%,1))
1120 IF (A9% > 96) AND (A9% < 123) THEN LET A9% = A9% - 32
1130 LET T2$ = T2$ + CHR$(A9%)
1140 NEXT K%
1150 IF T1$ > T2$ THEN SWAP A$(I%), A$(J%)
```

This ran extremely slowly, because string functions are very time consuming.

A look at the PTR function showed a faster way. The PTR returns the address of the variable, which in the case of string variables contains another pointer to the actual string and the length of the string. My BASIC code then could look like the following:

```
1000 LET T1$ = A$(I%)
1010 LET T2$ = A$(J%)
1020 LET A% = USR(PTR(T1$))
1030 LET A% = USR(PTR(T2$))
1040 IF T1$ > T2$ THEN SWAP A$(I%), A$(J%)
```

All that needs to be done is to set up the USR function to convert a string from lower case to upper case.

For 6800 BASIC, my USR function looked like this:

```
            ORG $A100

            LDX   $0026
            LDX   0,X
            STX   COUNT
            LDX   $0026
            LDX   0,X
LOOP        LDA B 0,X
            LDA A COUNT
            SUB B #1
            SBC A #0
            BCS   OVER
            LDA A 0,X
            CMP A #'z
            BHI   SKIP
            CMP A #'a
            BCS   SKIP
            SUB A #32
SKIP        STA A 0,X
            INX
            BRA   LOOP
OVER        RTS
COUNT       RMB   2

            END
```

I stored the USR function in data statements and then used a for-next loop to poke the machine language program into memory. Since I used the utility command space of FLEX, I had to be careful to avoid the EXEC command, and clobbering the USR code with a disk resident program.

6809 BASIC users should ORG their versions at $C100 and keep the string length count in the Y register.

```
            ORG   $C100

            LDX   [$CC32]   FLEX MEMEND
            LDY   2,X
            LDX   ,X
LOOP        CMPY  #0
            BEQ   OVER
            LDA   ,X+
            LBMI  -1,Y
            CMPA  #'z
            BHI   SKIP
            CMPA  #'a
            BCS   SKIP
            SUBA  #32
SKIP        STA   ,X
            BRA   LOOP
OVER        RTS
```

My 6809 code is a little hypothetical, since I'm one of those die-hards still running a 6800. Ask me instead about the M68K machine I'm building.

Very truly yours,

William R Hamblen

William R. Hamblen

Greetings,

68 Micro Journal is a fine magazine, one I can honestly say I look forward to each month. Thanks, and keep up the good work.

Computers are not my only interest (my wife wouldn't believe this!). I have been an amateur radio operator for more years than I choose to remember. Recently I have been noticing a lot more ads for good quality and inexpensive interfaces for RTTY and CW in the amateur radio literature. Unfortunately, the software is generally for VIC's, TRS's, IBM's and the like with no mention of SWTPC, Gimix or some of the other names familiar to 68 Micro readers.

I would like to hear from other "Hams" on this subject. Do you know of software available by name, and/or for hams? Not only RTTY and CW terminal programs, but other things useful in the ham shack.

If I get enough input, I will put together a catalog to distribute to others of like interest. CQ Hams with 68xx computers, I want to hear from you. 73'

Sincerely,

Jim Bertrand, WA9CIG

4522 N. 22 Street
Milwaukee, WI. 53209

IMPROVEMENTS? TO TSC "FILES" UTILITY

The FILES utility supplied by TSC provides a nice compact listing of all the files on a particular disk, however it does not indicate the number of free sectors remaining, or the name or number of the disk being processed. (See Example 1)

The latter is particularly important when generating hard copy listings of a number of disks, as trying to correlate "X" number of sheets of paper, each headed "FILES ON DRIVE NUMBER 1", with an equal number of disks, is a difficult task. The accompanying source listing is a version of the FILES utility which has been modified to overcome the above shortcomings. (See Example 2)

For those readers without the facilities to assemble the above source listing, an object code listing is provided. NOTE: The last pair of digits on each line is a checksum.

One other change not previously mentioned, is that the spacing between columns has been reduced by two. This allows five columns of files to fit on an ASR 33 printout, instead of the normal four. (TTYSET Width parameter must be set to the default value of 80 characters per line for this patch to work). Reversion to the original spacing can be achieved by changing the value at line 257 from 13 to 15, or the byte at memory location $A2D8 from $0D to $0F.

Also, the value at line 72, (memory location $A12C) which is the default column count, can be changed to 03 for those with PR-40 printers. This last change, together with the column spacing change above, allows a three column printout on a PR-40 or similar printer.

```
1.00=          OPT    PAG
2.00=          OPT    NOG
3.00= TTL MODIFIED "FILES" UTILITY
4.00=          PAG
5.00=*
6.00=*************************************
7.00=*  MODIFIED TSC "FILES" UTILITY      *
8.00=*          FOR FLEX 2                *
9.00=*       BY TERRY WALTERS             *
10.00=*                                   *
11.00=*  SHOWS VOLUME NAME AND NUMBER      *
12.00=*    OF DISK BEING PROCESSED        *
13.00=*   DATE OF CREATION OF DISK         *
14.00=*  AND NUMBER OF SECTORS FREE        *
15.00=*************************************
16.00=
17.00=* GLOBAL VARIABLES
18.00=
19.00=EOL      EQU     $AC02
20.00=WIDTH    EQU     $AC04
21.00=FCB      EQU     $A840
22.00=WASN     EQU     $AC0C
23.00=ASN      EQU     WASN
24.00=LSTTRM   EQU     $AC11
25.00=BUFPNT   EQU     $AC14
26.00=INDEX    EQU     $AC3D
27.00=DATENT   EQU     $AC41
28.00=SPC      EQU     $20
29.00=FMSCLS   EQU     $B403
30.00=FMS      EQU     $B406
31.00=WARMS    EQU     $AD03
32.00=GETCHR   EQU     $AD15
33.00=PUTCHR   EQU     $AD18
34.00=PSTRNG   EQU     $AD1E
35.00=CLASS    EQU     $AD21
36.00=PCRLF    EQU     $AD24
37.00=NXTCH    EQU     $AD27
38.00=GETFIL   EQU     $AD2D
39.00=OUTDEC   EQU     $AD39
40.00=OUTHEX   EQU     $AD3C
41.00=RPTERR   EQU     $AD3F
42.00=GETHEX   EQU     $AD42
43.00=INDEC    EQU     $AD4B
44.00=
45.00=         ORG     $A100
46.00=FIL      BRA     FIL1
47.00=VN       FCB     2              VERSION NUMBER
48.00=NAME     FCB     0,0,0,0
49.00=         FCB     0,0,0,0
50.00=EXT      FCB     0,0,0
51.00=ALLFLG   FCB     0
52.00=ALLDR    FCB     0
53.00=LOOP     FCB     0
54.00=POINT    FDB     0
55.00=COUNT    FCB     0
56.00=COL      FCB     0
57.00=NPL      FCB     0
58.00=DEC1     FDB     0
59.00=TEMP     FDB     0
60.00=
61.00=* PROGRAM STARTS HERE
62.00=
63.00=FIL1     LDX     BUFPNT         POINT TO BUFFER
64.00=         STX     POINT          SAVE POS
65.00=         LDA A   WIDTH          GET WIDTH
66.00=         BEQ     FIL12          SET?
67.00=         ASRA                   DIVIDE BY 16
68.00=         ASRA
69.00=         ASRA
70.00=         ASRA
71.00=         BRA     FIL15
72.00=FIL12    LDA A   #5             SET DEFAULT
73.00=FIL15    STA A   NPL            SET NUM PER LINE
74.00=         JSR     PCRLF          PRINT CR & LF
75.00=FIL2     TST     ALLDR          ALL DRIVES?
76.00=         BEQ     FIL22
77.00=FIL21    LDX     #FCB           POINT TO FCB
78.00=         LDA A   #20            FIND DRIVE
79.00=         STA A   0,X
80.00=         JSR     FMS            CALL FMS
81.00=         BNE     FIL27          ERROR?
82.00=         LDX     POINT          RESTORE POINTER
83.00=         STX     BUFPNT
84.00=         LDA A   0,X            GET CHARACTER
85.00=         STA A   LSTTRM         SET TERM
86.00=         BRA     FIL4
87.00=FIL22    LDX     POINT          GET POINTER
88.00=         LDA A   0,X            GET CHARACTER
89.00=         CMP A   #'.            IS IT PERIOD?
90.00=         BEQ     FIL25
91.00=         JSR     CLASS          CLASSIFY IT
92.00=         BCS     FIL25          TERM?
93.00=         CMP A   #'9            IS IT NUMBER?
94.00=         BLS     FIL3
```

```
 95.00=FIL25   TST      LOOP        LOOPING?
 96.00=        BEQ      FIL29
 97.00=FIL27   JMP      WARMS       RETURN TO FLEX
 98.00=FIL29   LDX      #FCB        POINT TO FCB
 99.00=        LDA A    ASN         GET DRIVE
100.00=        STA A    3,X         SET IN FCB
101.00=        BPL      FIL4
102.00=        STA A    ALLDR       SET ALL DRIVES
103.00=        BRA      FIL21
104.00=FIL3    LDX      PDINT       RESTORE POINTER
105.00=        STX      BUFPNT
106.00=        JSR      NXTCH       GET NEXT CHAR
107.00=        AND A    #3          MASK NUMBER
108.00=        LDX      #FCB        POINT TO FCB
109.00=        STA A    3,X         SAVE IN DRIVE
110.00=        JSR      NXTCH       GET NEXT CHAR
111.00=        BCS      FIL35       TERM?
112.00=        JMP      SERR        IF NOT, ERROR
113.00=FIL35   LDX      BUFPNT      SET POINTER
114.00=        STX      POINT
115.00=FIL4    LDX      #FCB        POINT TO FCB
116.00=        LDA A    #16         OPEN SYSTEM REC
117.00=        STA A    0,X
118.00=        JSR      FMS
119.00=        BNE      FIL43       ERROR?
120.00=        JSR      GETIR       GET FIRST RECORD
121.00=        BNE      FIL43       ERROR?
122.00=        LDX      #FILST      POINT TO STRING
123.00=        JSR      PSTRNG      PRINT IT
124.00=        LDX      #FCB+4      POINT TO NAME
125.00=        LDA B    #8          SET COUNT
126.00=        JSR      PDATA       PRINT NAME
127.00=        LDX      #NMST       POINT TO STRING
128.00=        LDA B    #2          SET COUNT
129.00=        JSR      PDATA       PRINT IT
130.00=        LDX      #FCB+15     POINT TO NUMBER
131.00=        JSR      OUTDEC      PRINT IT
132.00=        LDX      #VST        POINT TO STRING
133.00=        LDA B    #12         SET COUNT
134.00=        JSR      PDATA       PRINT IT
135.00=        LDX      #FCB+2      POINT TO DRIVE
136.00=        CLR      0,X
137.00=        CLR B
138.00=        JSR      OUTDEC      PRINT IT
139.00=        LDX      #FRST       POINT TO STRING
140.00=        JSR      PSTRNG      PRINT IT
141.00=        CLR B
142.00=        LDX      #FCB+21
143.00=        JSR      OUTDEC      PRINT FREE SECTORS
144.00=        LDX      #CSTR       POINT TO STRING
145.00=        LDA B    #13
146.00=        JSR      PDATA       PRINT IT
147.00=        CLR B
148.00=        LDX      #FCB+23     POINT TO DATE
149.00=        JSR      OUTDAT      PRINT IT
150.00=        JSR      PCRLF       PRINT CR & LF
151.00=FIL42   JSR      TSTTRM      TEST TERMINATOR
152.00=        BNE      FIL45
153.00=        CLR      ALLFLG      CLEAR FLAG
154.00=        BRA      FIL6
155.00=FIL43   JMP      ERR         REPORT ERROR
156.00=FIL45   LDX      BUFPNT      RESET POINTER
157.00=        LDA A    0,X         GET NEXT CHAR
158.00=        CMP A    #'.         IS IT PERIOD?
159.00=        BEQ      FIL5
160.00=        JSR      CLASS       CLASSIFY IT
161.00=        BCS      SERR        TERM?
162.00=        CMP A    #'9         IS IT NUMBER?
163.00=        BHI      FIL5
164.00=FIL47   JSR      NXTCH       GET NEXT
165.00=        BCC      FIL47       TERM?
166.00=        BRA      FIL42
167.00=FIL5    JSR      GETNAM      INPUT NAME
168.00=        BCS      SERR        ERROR?
169.00=FIL6    JSR      PRDIR       GO PRINT DIR
170.00=        BSR      TSTTRM      TEST TERM
171.00=        BNE      FIL5
172.00=        JSR      PCRLF       PRINT CR & LF
173.00=        JMP      FIL2        REPEAT
174.00=
175.00=* EXIT DIR UTILITY
176.00=
177.00=EXIT    JSR      PCRLF       PRINT CR & LF
178.00=        JMP      WARMS       RETURN TO FLEX
179.00=
180.00=* REPORT SYNTAX ERROR
181.00=
182.00=SERR    LDX      #FCB        POINT TO FCB
183.00=        LDA B    #26         SET IN ERROR
184.00=        STA B    1,X
185.00=
186.00=* REPORT DISK ERROR
187.00=
188.00=ERR     JSR      RPTERR      REPORT ERROR
189.00=ERR2    JMP      WARMS       RETURN TO FLEX
190.00=
191.00=* TEST FOR TERMINATOR
192.00=
193.00=TSTTRM  LDA A    LSTTRM      CHECK LAST TERM
194.00=        CMP A    #$0D        IS IT CR?
195.00=        BEQ      TSTTR2
196.00=        CMP A    EOL         IS IT EOL?
197.00=TSTTR2  RTS
198.00=
199.00=* PRINT DIRECTORY LIST
200.00=
201.00=PRDIR   CLR      COL
202.00=        LDX      #FCB        POINT TO FCB
203.00=        LDA A    #6          OPEN DIRECTORY
204.00=        STA A    0,X
205.00=        JSR      FMS
206.00=        BNE      ERR         ERRORS?
207.00=        INC      LOOP        SET LOOP FLAG
208.00=PRDIR2  JSR      PCRLF       PRINT CR & LF
209.00=PRDIR3  JSR      GETIR       GET INFO RECORD
210.00=        BEQ      PRDI32      ERRORS?
211.00=        JMP      PRDIR6
212.00=PRDI32  LDX      #FCB        POINT TO FCB
213.00=        TST      4,X         NAME PRESENT?
214.00=        BNE      PRDI34
215.00=        TST      COL         CHECK COLUMN
216.00=        BEQ      PRDI33
217.00=        JSR      PCRLF       PRINT CR & LF
218.00=PRDI33  JMP      PRDIR7
219.00=PRDI34  BMI      PRDIR3      DELETED NAME?
220.00=        LDA A    15,X        CHECK C PROT
221.00=        AND A    #$10
222.00=        BNE      PRDIR3
223.00=        LDX      #FCB+4      POINT TO NAME
224.00=        TST      ALLFLG      ALL FILES?
225.00=        BNE      PRDIR4
226.00=        STX      DATPNT      SAVE POINTER
227.00=        LDX      #NAME       TRY TO MATCH NAME
228.00=        STX      INDEX
229.00=PRDI35  LDX      INDEX
230.00=        CPX      #EXT+3
231.00=        BEQ      PRDIR4
232.00=        LDA A    0,X
233.00=        INX
234.00=        STX      INDEX
235.00=        LDX      DATPNT
236.00=        TST A                CHAR NULL?
237.00=        BEQ      PRDI37
238.00=        CMP A    0,X         DO COMPARISON
239.00=        BNE      PRDIR3
240.00=PRDI37  INX                  MOVE TO NEXT
241.00=        STX      DATPNT
242.00=        BRA      PRDI35
243.00=PRDIR4  LDX      #FCB+4      POINT TO NAME
244.00=        CLR      COUNT
245.00=        LDA B    #8          SET COUNTER
246.00=        JSR      PRNAM       PRINT NAME
247.00=        LDA A    #'.         OUTPUT PERIOD
248.00=        JSR      PUTCHR
249.00=        LDA B    #3          SET COUNT
250.00=        JSR      PRNAM       PRINT EXT
251.00=        INC      COL         BUMP COLUMN COUNT
252.00=        LDA A    COL         GET COUNT
253.00=        CMP A    #PL         MAX YET?
254.00=        BNE      PRDI45
255.00=        CLR      COL         CLEAR COUNT
256.00=        JMP      PRDIR2      REPEAT
257.00=PRDI45  LDA B    #13         FILL SPACES
258.00=        SUB B    COUNT
259.00=PRDIR5  LDA A    #$20        SETUP SPACE
260.00=        JSR      PUTCHR      OUTPUT IT
261.00=        DEC B                DEC THE COUNT
262.00=        BNE      PRDIR5
263.00=        JMP      PRDIR3
264.00=PRDIR6  LDA A    1,X         GET ERROR TYPE
265.00=        CMP A    #8          IS IT EOF?
266.00=PRDI65  BEQ      PRDIR7
267.00=        JMP      ERR         REPORT ERROR
268.00=PRDIR7  RTS
269.00=
270.00=* OUTPUT SPACE
271.00=
272.00=OUTSP   LDA A    #$PC        SETUP SPACE
273.00=        JMP      PUTCHR      PRINT IT
274.00=
275.00=* GET INFO RECORD
276.00=
277.00=GETIR   LDX      #FCB        POINT TO FCB
278.00=        LDA A    #7          GET IR CODE
279.00=        STA A    0,X
280.00=        JMP      FMS         CALL FMS
281.00=
```

```
282.00=* PRINT FILE NAME
283.00=
284.00=PRNAM   LDA A   0,X          GET CHARACTER
285.00=        BEQ     PRNAM2
286.00=        JSR     PUTCHR       OUTPUT CHAR
287.00=        INC     COUNT        BUMP COUNT
288.00=PRNAM2  INX                  BUMP TO NEXT
289.00=        DEC B
290.00=        BNE     PRNAM
291.00=        RTS                  RETURN
292.00=
293.00=* GET FILE MATCH NAME
294.00=
295.00=GETNAM  CLR     ALLFLG       CLEAR ALL
296.00=        LDX     #NAME        POINT TO BUFFER
297.00=        LDA B   #11          SET COUNT
298.00=GETNA2  CLR     0,X          CLEAR BUFFER
299.00=        INX
300.00=        DEC B
301.00=        BNE     GETNA2
302.00=        LDX     BUFPNT       POINT TO NAME
303.00=        LDA A   0,X          GET CHARACTER
304.00=        CMP A   #'.          IS IT '.'?
305.00=        BED     GETNA4
306.00=        LDX     #NAME-4      RESET POINTER
307.00=        JMP     GETFIL       GET FILE NAME
308.00=GETNA4  INC     BUFPNT+1     FIX POINTER
309.00=        LDA B   #3           SET COUNT
310.00=        LDX     #EXT         POINT TO EXT
311.00=GETNA5  JSR     NXTCH        GET CHARACTER
312.00=        BCS     GETNA6       TERM?
313.00=        STA A   0,X          SAVE CHAR
314.00=        INX
315.00=        DEC B
316.00=        BPL     GETNA5
317.00=        SEC
318.00=        RTS                  RETURN
319.00=GETNA6  CLC
320.00=        RTS
321.00=
322.00=* PRINT STRING WITHOUT CR & LF
323.00=
324.00=PDATA   LDA A   0,X          GET CHARACTER
325.00=        BEQ     PDATA2       NULL?
326.00=        JSR     PUTCHR       PRINT IT
327.00=        INX                  BUMP TO NEXT
328.00=        DEC B                DEC COUNT
329.00=        BNE     PDATA
330.00=        RTS                  RETURN
331.00=PDATA2  CLR B
332.00=        RTS
333.00=
334.00=* OUT DATE DIGIT
335.00=OUTD    LDX     #DEC1        POINT TO DIGIT
336.00=        STA A   1,X          SAVE CHAR
337.00=        CMP A   #9           LESS THAN 10?
338.00=        BHI     OUTD2
339.00=        JSR     OUTSP        DO SPACE
340.00=OUTD2   CLR B
341.00=        JMP     OUTDEC       PRINT DIGITS
342.00=
343.00=* OUTPUT DATE
344.00=
345.00=OUTDAT  STX     TEMP         SAVE POINTER
346.00=        LDA A   1,X          GET DAY
347.00=        BSR     OUTD         PRINT IT
348.00=        BSR     OUTM
349.00=        LDX     TEMP
350.00=        LDA A   0,X          GET MONTH
351.00=        LDX     #MNTH        POINT TO MONTHS
352.00=OUTDA4  DEC A
353.00=        BEQ     OUTDA5
354.00=        INX                  FIND IN TABLE
355.00=        INX
356.00=        INX
357.00=        BRA     OUTDA4
358.00=OUTDA5  LDA B   #3           SET COUNT
359.00=        JSR     PDATA        PRINT MONTH
360.00=        BSR     OUTM
361.00=        LDX     TEMP         RESET POINTER
362.00=        LDA A   2,X          GET YEAR
363.00=        BRA     OUTD         PRINT IT
364.00=OUTM    LDA A   #'-          SETUP '-'
365.00=        JMP     PUTCHR       PRINT IT
366.00=
367.00=* STRINGS
368.00=
369.00=FILST   FCC     'FILES ON DISK '
370.00=        FCB     4
371.00=CSTR    FCC     '    CREATED: '
372.00=FRST    FCC     'SECTORS FREE = '
373.00=        FCB     4
374.00=VST     FCC     '" IN DRIVE #'
375.00=NMST    FCC     ' #'
376.00=MNTH    FCC     'JANFEBMARAPRMAYJUN'
377.00=        FCC     'JULAUGSEPOCTNOVDEC'
378.00=        END     FIL
```

Example 1. Header produced by the FILES Utility supplied by TSC.

```
FILES ON DRIVE NUMBER 1

COPY.CMD        DIR.CMD         COPY9.TXT       FILE1.TXT       FILE1.CMD
F.CMD           LIFECON.TXT     LIFECON.CMD     FR.TXT          FR.CMD
GET.TXT         PET.CMD         FILELETR.TXT    MPASCAL.BIN     LETTER.OBJ
PRUL.TXT        PRUL.CMD        PRIMES.TXT      BC-STN2.TXT     BC-STN.TXT
BC-STN1.TXT     LETTER.TXT
```

Example 2. Header produced by accompanying program.

```
FILES ON DISK "WORK #4on2" IN DRIVE #1
SECTORS FREE = 161    CREATED: 12-SEP-81

COPY.CMD        DIR.CMD         COPY9.TXT       FILE1.TXT       FILE1.CMD
F.CMD           LIFECON.TXT     LIFECON.CMD     FR.TXT          FR.CMD
GET.TXT         GET.CMD         FILELETR.TXT    MPASCAL.BIN     LETTER.OBJ
PRUL.TXT        PRUL.CMD        PRIMES.TXT      BC-STN2.TXT     BC-STN.TXT
BC-STN1.TXT     LETTER.TXT
```

```
$113 A100 20 1B 02 00 00 00 00 00 00 00 00 00 00 00 00 11
$113 A110 00 00 00 00 00 00 00 00 00 00 FE AC 14 FF A1 11 CC
$113 A120 26 AC 04 27 06 47 47 47 47 20 02 86 05 B7 A1 15 62
$113 A130 BD AD 24 7D A1 0F 27 19 CE A8 40 86 14 A7 00 BD 6C
$113 A140 04 06 26 24 FE A1 11 FF AC 14 A6 00 B7 AC 11 20 5E
$113 A150 47 FE A1 11 A6 00 81 2E 27 09 BD AD 21 25 04 81 4A
$113 A160 39 23 17 7D A1 10 27 03 7E AD 03 CE A8 4B B6 AC 0A
$113 A170 0C A7 03 2A 23 B7 A1 0F 20 BE FE A1 11 FF AC 14 24
$113 A180 BD AD 27 04 03 CE A8 40 A7 03 BD AD 27 25 03 7E 1C
$113 A190 A2 33 FE AC 14 FF A1 11 CE A8 40 86 1B A7 00 BD C7
$113 A1A0 04 06 26 5B BD A2 F6 26 56 CE A3 BD BD AD 1E CE 4B
$113 A1B0 A8 44 C6 08 BD A3 43 CE A3 C6 C6 02 BD A3 43 CE CE
$107 A1C8 A8 4F BD AD 36
$113 A1C4 39 CE A3 8A C6 0C BD A3 43 CE A8 42 4F 00 5F BD 6B
$113 A1D4 AD 39 CE A3 AA BD AD 1E 5F CE A8 55 BD AD 39 CE 53
$113 A1E4 A3 9D C6 BD BD A3 43 5F CE A8 97 BD A3 61 BD AD 3A
$113 A1F4 24 BD A2 40 26 08 7F A1 0E 20 21 7E A2 3A FE AC F3
$113 A204 14 A6 00 81 2E 27 10 BD AD 21 25 23 81 39 22 07 F0
$113 A214 BD AD 27 24 F8 20 0A BD A3 0F 25 13 BD A2 48 BD AE
$113 A224 18 26 F4 BD AD 24 7E A1 33 BD AD 24 7E A8 03 CE 07
$113 A234 A8 48 C6 1A E7 01 BD AD 3F 7E AD 03 86 AC 11 81 9B
$113 A244 0D 27 03 81 AC 02 39 7F A1 14 CE A8 40 86 06 A7 1A
$113 A254 00 BD 84 06 26 E0 7C A1 10 BD AD 24 BD A2 F6 27 42
$113 A264 03 7E A2 E7 CE A8 40 A6 04 26 0B 7D A1 14 27 03 2B
$113 A274 BD AD 24 7E A2 F0 2B E4 A6 3F 84 18 26 DE CE A8 66
$107 A284 44 7D A1 0E 62
$113 A288 26 27 FF AC 41 CE A1 03 FF AC 3D FE AC 3D BC A1 1B
$113 A298 0E 27 16 A6 00 08 FF AC 3D FE AC 41 4D 27 04 A1 C0
$113 A2A8 00 26 B5 08 FF AC 41 20 E2 CE A8 44 7F A1 13 C6 1E
$113 A2B8 38 BD A3 00 96 2E BD AD 1B C6 03 BD A3 00 7C A1 AE
$113 A2C8 14 86 A1 14 81 A1 15 26 06 7F A1 14 7E A2 5D C6 F9
$113 A2D8 0D F0 A1 13 86 20 BD AD 1B 5A 26 F8 7E A2 60 A6 F8
$113 A2E8 01 81 08 27 03 7E A2 3A 39 86 28 7E AD 1B CE A8 BC
$113 A2F8 40 86 07 A7 00 7E 84 06 A6 00 27 06 BD AD 1B 7C D3
$113 A308 A1 13 08 5A 26 F2 39 7F A1 0E CE A1 03 C6 08 6F FA
$113 A318 00 00 5A 26 FA FE AC 14 A6 08 81 2E 27 06 CE A0 01
$113 A328 FF 7E AD 2D 7C AC 15 C6 03 CE A1 08 BD AD 27 23 94
$113 A338 08 A7 00 08 5A 2A F5 0D 39 0C 39 A6 00 27 08 BD C4
$107 A340 AD 1B 08 5A E6
$113 A34C 26 F5 39 3F 39 CE A1 16 A7 81 81 09 22 03 BD A2 D6
$113 A35C F1 3F 7E AD 39 FF A1 1B A6 01 BD E9 BD 1E FE A1 1A
$113 A36C 1B A6 00 C8 A3 C8 4A 27 05 08 08 08 20 F8 C6 03 77
$113 A37C BD A3 43 BD B7 FE A1 10 A6 02 2B C9 86 2D 7E AD 70
$113 A38C 1B 46 49 4C 45 53 20 4F 4E 20 44 49 53 4B 20 22 E8
$113 A39C 04 20 20 20 20 43 52 45 41 54 45 44 3A 20 53 45 3F
$113 A3AC 43 54 4F 52 53 20 46 52 45 45 20 3D 20 04 22 20 0D
$113 A3BC 49 4E 20 44 52 49 56 45 20 23 20 23 4A 41 4E 46 B7
$113 A3CC 45 42 4D 41 52 41 50 52 4D 41 59 4A 55 4E 4A 55 C0
$113 A3DC 4C 41 55 47 53 45 50 4F 43 54 4E 4F 56 44 45 43 B7
59
```

Flex user Notes listings continued from page 11

```
1000 00 01
                    *
                    *
                    * REAL VARIABLES
                    *
100C                HEIGHT RMB   6
1012                RADIUS RMB   6
1018                WEIGHT RMB   6
                    *
                    * CALCULATION OF WEIGHT ASSUMES THAT HEIGHT AND
                    * RADIUS OF CYLINDER HAVE BEEN CALCULATED OR ENTERED
                    * AND ARE AVAILABLE IN THE ABOVE TWO VARIABLE LOCATIONS
                    *
                    * THESE ASSUME THE EXISTENCE OF A MATH PACKAGE AND A
                    * ROUTINE 'MOVE6' THAT MOVES SIX BYTES FROM LOCATION
                    * POINTED AT BY I TO LOCATION POINTED AT BY Y
                    *
```

```
0000                    ORG   0         PROGRAM ORIGIN
0000 8E  1000  CALCWT   LDX   #PI
0003 10BE 3006          LDY   #XSIGN
0007 BD  2015           JSR   MOVE6
000A 8E  1012           LDX   #RADIUS
000D 10BE 300C          LDY   #YSIGN
0011 BD  2015           JSR   MOVE6
0014 BD  2000           JSR   FPMUL     MULTIPLY, RESULT IN RSIGN  PI*R
>0017 BD  0052          JSR   RTOI
001A 8E  1012           LDX   #RADIUS
001D 10BE 300C          LDY   #YSIGN
0021 BD  2015           JSR   MOVE6
0024 BD  2000           JSR   FPMUL     PI * R^2
>0027 BD  0052          JSR   RTOI
002A 8E  1006           LDX   #RHO
002D 10BE 300C          LDY   #YSIGN
0031 BD  2015           JSR   MOVE6
0034 BD  2000           JSR   FPMUL     PI * R^2*RHO
>0037 BD  0052          JSR   RTOI
003A 8E  100C           LDX   #HEIGHT
003D 10BE 300C          LDY   #YSIGN
0041 BD  2015           JSR   MOVE6
0044 BD  2000           JSR   FPMUL     PI * R^2 * RHO * HEIGHT
0047 8E  3000           LDX   #RSIGN
004A 10BE 1018          LDY   #WEIGHT
004E BD  2015           JSR   MOVE6     PUT RESULT IN WEIGHT
0051 39                 RTS
                        *
                        * SUBROUTINE RTOI
                        *
0052 8E  3000  RTOI     LDX   #RSIGN
0055 10BE 3006          LDY   #XSIGN
0059 7E  2015           JMP   MOVE6
           005C BYTECT  EQU   *         BYTE COUNT FOR CALC
                        *
                        END

0 ERROR(S) DETECTED
```

```
                        *
                        * PROGRAM TO CALCULATE WT OF CYLINDER USING BINARY
                        * MATH PACKAGE WITH INTERPRETER
                        *
                        * MATH PKG EQUATES
                        *
            0300  MATH  EQU   $300
                        *
                        * MATH FUNCTION MNEMONICS EQUATES
                        *
            0001  MUL   EQU   $01
            0007  PSH   EQU   $07
            000B  STR   EQU   $0B
                        *
1000                    ORG   $1000
                        * CONSTANTS
                        *
1000 00 00 00 00  PI    FCB   $00,$00,$00,$00
1004 00 00 00 00  RHO   FCB   $00,$00,$00,$00
                        *
                        * VARIABLES
                        *
1008            RADIUS  RMB   6
100E            HEIGHT  RMB   6
1014            WEIGHT  RMB   6
                        *
0000                    ORG   0
                        *
0000 9D  0300  CALCWT   JSR   MATH
0003 1000               FDB   PI
0005 07                 FCB   PSH
0006 1008               FDB   RADIUS
0008 07                 FCB   PSH
0009 01                 FCB   MUL
000A 1008               FDB   RADIUS
000C 07                 FCB   PSH
000D 01                 FCB   MUL
000E 100E               FDB   HEIGHT
0010 07                 FCB   PSH
0011 01                 FCB   MUL
0012 1004               FDB   RHO
0014 07                 FCB   PSH
0015 0001               FDB   MUL
0017 1014               FDB   WEIGHT
0019 0B                 FCB   STR
001A 39                 RTS
           0018 BYTECT  EQU   *
                        END

0 ERROR(S) DETECTED
```

```
                        *
                        * CALCULATE THE WEIGHT OF A STEEL CYLINDER USING
                        * TSC BCD MATH PACKAGE ASSUMING 6809 VERSION AVAILABLE
                        * THIS VERSION WITH MACROS NOT EXPANDED
                        *
                        * MATH PKG EQUATES
            2000  FPMUL EQU   $2000
            2015  MOVE6 EQU   $2015
            3000  RSIGN EQU   $3000
            3006  XSIGN EQU   $3006
            300C  YSIGN EQU   $300C
                        *
                        * MACRO DEFINITIONS
                        *
                FMUL    MACRO
                        IFC   &1,*
                        JSR   RTOI
                        ELSE
                        LDX   #&1
                        LDY   #XSIGN
                        JSR   MOVE6
                        ENDIF
                        LDX   #&2
                        LDY   #YSIGN
                        JSR   MOVE6
                        JSR   FPMUL
                        ENDM
                        *
                STORE   MACRO
                        LDX   #RSIGN
                        LDY   #&1
                        JSR   MOVE6
                        ENDM
                        *
                        * CONSTANTS
                        *
1000                    ORG   $1000
                        *
1000 00 03 14 15  PI    FCB   $00,$03,$14,$15,$92,$01 SIGN, 7 DIGITS, EXP
1004 92 01
1006 00 04 52 80  RHO   FCB   $00,$04,$52,$80,$00,$01 DENSITY OF STEEL
100A 00 01
                        *
                        *
                        * REAL VARIABLES
                        *
100C            HEIGHT  RMB   6
1012            RADIUS  RMB   6
1018            WEIGHT  RMB   6
                        *
                        * CALCULATION OF WEIGHT ASSUMES THAT HEIGHT AND
                        * RADIUS OF CYLINDER HAVE BEEN CALCULATED OR ENTERED
                        * AND ARE AVAILABLE IN THE ABOVE TWO VARIABLE LOCATIONS
                        *
                        * THESE ASSUME THE EXISTENCE OF A MATH PACKAGE AND A
                        * ROUTINE "MOVE6" THAT MOVES SIX BYTES FROM LOCATION
                        * POINTED AT BY X TO LOCATION POINTED AT BY Y
                        *
0000                    ORG   0         PROGRAM ORIGIN
                        *
0000            CALCWT  FMUL  PI,RADIUS
0017                    FMUL  *,RADIUS
0027                    FMUL  *,RHO
0037                    FMUL  *,HEIGHT
0047                    STORE WEIGHT
0051 39                 RTS
                        *
                        * SUBROUTINE RTOI
                        *
0052 8E  3000  RTOI     LDX   #RSIGN
```

```
0055 108E 3008           LDY   #YSIGN
0059 7E   2015           JMP   MOVE6
                 005C BYTECT EQU   *        BYTE COUNT FOR CALC
                          *
                          END

0 ERROR(S) DETECTED
                          *
                          * CALCULATE THE WEIGHT OF A STEEL CYLINDER USING
                          * TSC BCD MATH PACKAGE ASSUMING 6809 VERSION AVAILABLE
                          * THIS VERSION WITH MACROS EXPANDED
                          *
                          * MATH PKG EQUATES
                  2000 FPMUL  EQU   $2000
                  2015 MOVE6  EQU   $2015
                  3000 RSIGN  EQU   $3000
                  3006 XSIGN  EQU   $3006
                  300C YSIGN  EQU   $300C
                          *
                          * MACRO DEFINITIONS
                          *
                          FMUL   MACRO
                                 IFC   &1,0
                                 JSR   RTOI
                                 ELSE
                                 LDI   #&1
                                 LDY   #XSIGN
                                 JSR   MOVE6
                                 ENDIF
                                 LDI   #2
                                 LDY   #YSIGN
                                 JSR   MOVE6
                                 JSR   FPMUL
                                 ENDM
                          *
                          STORE  MACRO
                                 LDI   #RSIGN
                                 LDY   #1
                                 JSR   MOVE6
                                 ENDM
                          *
                          * CONSTANTS
                          *
1000                             ORG   $1000
                          *
1000 09 03 14 15          PI     FCB   $00,$03,$14,$15,$92,$01 SIGN, 7 DIGITS, EXP
1004 92 01
1006 00 04 52 80          RHO    FCB   $00,$04,$52,$80,$00,$01 DENSITY OF STEEL
100A 00 01
                          *
                          *
                          * REAL VARIABLES
                          *
100C                      WEIGHT RMB   6
1012                      RADIUS RMB   6
1018                      HEIGHT RMB   6
                          *
                          * CALCULATION OF WEIGHT ASSUMES THAT HEIGHT AND
                          * RADIUS OF CYLINDER HAVE BEEN CALCULATED OR ENTERED
                          * AND ARE AVAILABLE IN THE ABOVE TWO VARIABLE LOCATIONS
                          *
                          * THESE ASSUME THE EXISTENCE OF A MATH PACKAGE AND A
                          * ROUTINE "MOVE6" THAT MOVES SIX BYTES FROM LOCATION
                          * POINTED AT BY X TO LOCATION POINTED AT BY Y
                          *
0000                             ORG   0        PROGRAM ORIGIN
                          *
0000                      CALCWT FMUL  PI,RADIUS
0000 8E   1000                   LDI   #PI
0003 108E 3006                   LDY   #XSIGN
0007 BD   2015                   JSR   MOVE6
000A 8E   1012                   LDI   #RADIUS
000D 108E 300C                   LDY   #YSIGN
0011 BD   2015                   JSR   MOVE6
0014 BD   2000                   JSR   FPMUL
                                 ENDM
0017                             FMUL  *,RADIUS
>0017 BD   0052                  JSR   RTOI
001A 8E   1012                   LDI   #RADIUS
001D 108E 300C                   LDY   #YSIGN
0021 BD   2015                   JSR   MOVE6
0024 BD   2000                   JSR   FPMUL
                                 ENDM
0027                             FMUL  *,RHO
>0027 BD   0052                  JSR   RTOI
002A 8E   1006                   LDI   #RHO
002D 108E 300C                   LDY   #YSIGN
0031 BD   2015                   JSR   MOVE6
0034 BD   2000                   JSR   FPMUL
                                 ENDM
0037                             FMUL  *,HEIGHT
>0037 BD   0052                  JSR   RTOI
003A 8E   100C                   LDI   #HEIGHT
003D 108E 300C                   LDY   #YSIGN
0041 BD   2015                   JSR   MOVE6
0044 BD   2000                   JSR   FPMUL
                                 ENDM
0047                             STORE WEIGHT
0047 8E   3000                   LDI   #RSIGN
004A 108E 1018                   LDY   #WEIGHT
004E BD   2015                   JSR   MOVE6
                                 ENDM
0051 39                          RTS
                          *
                          * SUBROUTINE RTOI
                          *
0052 8E   3000            RTOI   LDI   #RSIGN
0055 108E 3006                   LDY   #XSIGN
0059 7E   2015                   JMP   MOVE6
                 005C BYTECT EQU   *        BYTE COUNT FOR CALC
                          *
                          END

0 ERROR(S) DETECTED
```

CLASSIFIED ADVERTISING

TELETYPE Model 43 PRINTER - with serial (RS232) interface, and full ASCII keyboard. LIKE NEW - New cost $1295.00 - ONLY $759.00 ready to run - Call Tom - Larry - Bob, CPI 615 842-4600

***

FOR SALE: Gimix #58 Disk Controller, single density, 5" and 8" drives, programmed I/O, with boot ROM and documentation. $95.
Phil Nunn, 201 Netherfield, Comstock Park, MI. 49321 tel: 616-784-5732

***

Wanted SWTPC Boards, MP-A2, MP-S, MP-L, MP-R For 6800. Other Brands I/O O.K.
J. Fiorino, 518-85 St. Brookyn, NY 11209

***

For Sale: Mint condition Gimix 6800 System. Loaded. Two Siemans disk drives, diskettes, RCA keyboard, Ball monitor, Best Offer, Complete system.
Call Bill 516-331-2046, work 516-575-2222

***

FOR SALE: GIMIX 6800 SS-50 MBoard with 6800 CPU, 56K RAM, 64 X 16 Video, BFD68 Diskcon.
Call Alex 201-235-3089, 9-4

***

For Sale: SWTPC 69/A. 6809, 28k RAM, Serial, Parallel, DS/DD disk controller, Flex09, Assembler, Editor $600. Epson MX-80 with Graftrax $350. Must sell.
Robert Alexander, 1630 Worcester Rd. Apt 222C, Framingham, Ma 01701, 617-872-6739 evenings.

***

For Sale: SSB VAR 6809 computer with 512KB RAM, 20MB Winchester, 750KB floppy, 2 Qume terminals, Okidata 2410 printer (350 cps), Brother HR-25 letter quality printer(25cps), OS9-L2, Microware C, Stylograph, Mailmerge, Spell, and modem software. Paid $16,000 in December. Buy for $9500 or make offer (on all or part). System barely used. Works like new. All manuals and boxes included.
Call 415-221-3333 ext 337 (California) and leave message. I'll call back ASAP.

***

FOR SALE: SWTPC MF-68:MP-09, 65k w.MP-8M&Dig.Res.&DSD, MP-S, MP-L, MP-T, MP-R, DC-1&MF-68 Disk, AC-30, PR-40 Prtr Newtech Music Brd, Compuware Cal/Clock, GT-6144, Much Software. $600.
Bob Stafford, 201 Saint Piere, Worthington, Oh 43085

***

ASSEMBLERS

Southeast Media

ASTRUK09

A "Structured Assembler for the 6809" which requires the TSC Macro Assembler. Allows direct use of structured statements such as IF, ELSE, DO, REPEAT, etc., and provides indented level formatting of the listing so that the structure is apparent. Re. '68' Micro Journal, Sept. '83 (program was called "STASM09"; has been renamed due to conflicts).

A User reports

"... I'm very pleased and am now writing almost exclusively in (ASTRUK09). I've selected it over --- for all future systems development... As (one) of my early evaluations, I rewrote a rather elaborate routine originally done in assembly. Out of the 1000 bytes of code generated, the (ASTRUK09) version used only 20 more bytes than the original. --- could not handle this program since it uses triple-precision fixed point arithmetic... I have a large body of code already written that is incompatible with --- constructs. No problem with (ASTRUK09) and the structure sure helps in understanding the logic!"

F, CCF - **$99.95**

TSC

Macro Assembler

The FLEX STANDARD Assembler. F,CCF $50.00

Relocating Assembler w/Linking Loader

Use with many of the C and Pascal Compilers. F,CCF $150.00

Great Plains Comp. Co.

RMAC

Relocating, Recursive Macro Assembler and **Linking Loader.**

F,CCF $120.00; w/Source $240.00

OmegaSoft

- RALLI

Relocating Assembler and Linking Loader

F,CCF $125.00; for One Year Maint., add $50.00

Windrush Micro Systems

MACE, by Graham Trott. F,CCF - $98.00

Computer Systems Consultants

SUPER SLEUTH

Computer Systems Consultants Super Sleuth is a "Time Tested", reliable, PROVEN Disassembler that has gained acceptance through out the SS-50 Bus Community as an extremely POWERFUL, INTERACTIVE, Software Tool. The Super Sleuth Software Package consists of 3 Programs; SLEUTH (the Disassembler), CMGNAM (used to globally Change Labels to a meaningful Name), and XREF (a Cross Reference Generator for Source Code Files). SLEUTH will Disassemble Memory Resident 6809 Code and 6800, 6801, 6802, 6803 (the "Baby CoCo"), 6805, 6808, 6809, and 6502 (Apple, Atari, Commodore, etc.) Binary Disk Files. (See Aug. '83 '68' Micro Journal "Color Users Notes" Column for a full Review.)

Color Computer	SS-50 Bus (all w/ Source)
CCO (32K Req'd)	
Obj. Only $49.00	F, $99.00
CCF, Obj. Only $50.00	U, $100.00
CCF, w/Source $99.00	O, $101.00
CCO, Obj. Only $50.00	

ALL Computer Systems Consultants Software runs on the Color FLEX Systems
ALL in stock
call [illegible]
for IMMEDIATE DELIVERY

Computer Systems Center

CYMBUTZ +

An "easy to use", powerful Disassembler for Disk Resident 6809 and 6800 Binary Files. Allows the development of a "Control File" of various Program "Boundaries" during successive disassemblies; can use a Label File which automatically replaces a Hex location with a Label Name; includes an XREF Utility; etc. Label Files provided for Mini-FLEX, FLEX2, FLEX9, Color Computer (for use with Color FLEX Systems), etc. OS-9 Version includes special OS-9 options.

CCF, Obj. Only $[illegible]
CCO, " " $150.00
F, " " $[illegible]
O, " " $150.00
U, " " $[illegible]

COMPILERS & DECOMPILERS

6809 "Structured" Assembly Lang. Compilers

Windrush Micro Systems

PL/9

By Graham Trott. A "Structured" Assembly Language Editor/Compiler/Debugger, all in ONE PACKAGE: provides a totally INTERACTIVE Program Development Cycle. The Compiler supports large Symbol Names, Variable Types, Pointers, Control Structures, Stack, A-, B-, and D-Register manipulation, etc. The Source-Oriented Trace/Debugger provides Single Stepping, Breakpointing, etc. An excellent Software Development Tool for utilizing the power of the 6809 in developing small to medium sized packages.

F, CCF - $198.00

Whimsical Developments

WHIMSICAL

Need the Ease of Design and Maintainability of "Structured Programming" AND the Speed and Control of Assembly Language? Then WHIMSICAL was designed for you! This Single Pass, Recursive Descent Compiler provides the tool for developing simple Utilities to MAJOR Systems in Assembly Language. Supports 3 "Lex" Levels which allow one level of Procedure nesting, or more within "Modules". It is easy to develop programs written for other machines since you are working at the Assembly Language level. Features unified, user-defined I/O; produces ROMable, relocatable, recursive, re-entrant Code; Structured style and statements with Procedures and Modules; supports Byte and Double-Byte primitives with 3 types of Integers (up to 32 bit), Char and Boolean, and unlimited sized Arrays (vectors only); Interrupt handling; unlimited length Variable Names; Variable Initialization (defaults to $00); Include "Source File" directive; Conditional compiling; direct Code insertion; control of the Stack Pointer; etc. To quote Ron Anderson in his review of WHIMSICAL in the Sept. '83 issue of '68' Micro Journal that, except for the lack of floats, "..., I have to give this one VERY high rating, ...". It is a FAST Compiler which produces FAST Code (his "Primes" Benchmark ran at 9 secs. on a 2 MHz System).

F and CCF - $195.00

'C' Compilers

Windrush Micro Systems

C Compiler

By James McCosh. Full featured C Compiler for the FLEX Operating System. Includes a Reloc. Assb., but needs the TSC Relocating Assembler/Linking Loader (which includes a Library Manager) for those "full blown" System Packages.

F and CCF - $295.00

Introl

C Compiler

A full-featured C, streamlined for the 6809. Generates very efficient object code. Output "benchmarks" close to 10MHz 68000 in 8 Bit Operations; 1.5 times faster than a 4 MHz Z80 when using a 2MHz 6809 System (Re. p 43, '68' Micro Journal, May '83). Floats, etc.

F, CCF, and O - $375.00
U - $425.00
One Year Maint. - $100.00

VISA MasterCard

"FLEX is a trademark of Technical Systems Consultants
"OS9 is a trademark of Microware

TOLL FREE
1-800-338-6800
For Ordering

SOUTH EAST MEDIA

5900 Cassandra Smith Rd.
Hixson, TN 37343
info (615) 842-4601

CoCo OS-9™ FLEX™
SOFTWARE

Availability Legends —
F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

PASCAL Compilers

TSC

PASCAL Compiler

Native Code Compiler (UCSD Oriented).

F and CCF - $200.00

Lucidata

PASCAL Compiler

P-Code Compiler (ISO Standard). Designed especially for Microcomputer Systems: Run-time System checks available resources for each task, allowing operation on even minimal computer systems. Allows linkage to Assembler Code for maximum flexibility.

F and CCF 5" - $198.00
F 8" - $285.00

OmegaSoft

PASCAL Compiler

For the PROFESSIONAL; ISO Based, Native Code Compiler. Primarily for Real-Time and Process Control applications. Use custom I/O devices in place of the Pascal INPUT and OUTPUT; Long Int. (32 Bit); Dynamic length strings; Interrupt processing, ROM-able, PIC, Re-Entrant Code, etc. [illegible] Includes Source for the Symbolic Debugger, Runtime, and several Utilities. Requires a "Motorola Compatible" Relocating Assembler and Linking Loader.

F and CCF - $425.00
One Year Maint. - $100.00

DECOMPILERS

Southeast Media

DUB (A UniFLEX "basic" De-Compiler)

Re-Create a Source Listing from UniFLEX Compiled basic Programs. Easy to Use; works w/ ALL Versions of UniFLEX basic; Output to Disk or Terminal. Time TESTED and PROVEN; SOLID!

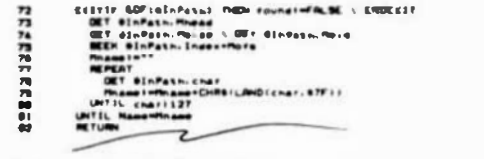
U - $219.95

UTILITIES

Southeast Media

BasicO9 XRef

This BasicO9 Cross Reference Utility is a BasicO9 Program which will produce a "pretty printed" listing with each line numbered, followed by a complete cross referenced listing of all variables, external procedures, and line numbers called. Also included is a Program List Utility which outputs the listing without the overhead of building the cross reference table, which allows it to run considerably faster when only a "pretty printed" listing with line numbers is desired. Requires BasicO9 or Run8 for operation.



O and CCO - Obj. Only -- $39.95
O and CCO - w/ Source - $79.95

Southeast Media

OS-9 VDisk

Give your OS-9 Level I System the speed of memory access that can be several orders of magnitude over your present floppy disk drive. Use that Extended Memory capability of your SWTPC or Gimix CPU card (or any other that has the same format DAT). The size of the Virtual Disk is completely variable in whole increments of 4K up to 960K, which is all that these systems can address beyond the base page that OS-9 Level I uses. By putting all of your CMDS Directory on your Virtual Disk, you can have the fastest execution speed possible (next to eating up System Memory with all of them). You can also set up high speed inter-process communications via random virtual disk files and not eat up valuable system memory with pipe buffers. Some Assembly Required - Level I ONLY.

O, obj. only - $79.95
w/ Source - $149.95

Southeast Media

O-F

---- OS/9 to FLEX - FLEX to OS/9 ----

Finally; the barrier has been removed between OS/9 and FLEX formatted disks! Now you can READ from, and WRITE to, a Single Sided 5" or 8" FLEX diskette from OS-9 with O-F. O-F is a new and unique program, written in BASICO9 (with Source), that performs the following functions;

REFORMAT: A BASICO9 Program that reformats a chosen amount of an OS-9 disk to FLEX Format so it can be used normally by FLEX.

FLEX: A BASICO9 Program that does the actual read or write function to the special O-F Transfer Disk, all selected from a user-friendly menu. Functions provided include reading the FLEX Directory, Deleting FLEX Files, Copying both directions, etc. All selections are interactive and complete, including all necessary prompts to the operator.

FLEX users can read, write and use the special disk as any other FLEX disk, provided the FLEX directory is not allowed to continue beyond track zero (too many files).

F and CCF - $79.95

Southeast Media

COPYMULT

--- Copy LARGE Disks to several smaller disks ---

The following FLEX utilities allow the backup of ANY size disk to any SMALLER size diskettes (Winchester to 8's or 5's, 8" to 5's, etc.). By simply inserting diskettes as requested by COPYMULT, a large disk system may be downloaded to your present floppy disk system, any size. No need to fiddle with directory deletions or any of the other tedious operations that must be done using the normal copy routines.

COPYMULT.CMD understands normal "copy" syntax and always keeps up with files already copied by maintaining directories for both host and receiving disk system, eliminating hours of tedious keyboard entries and other time consuming cleanup chores.

BACKUP.CMD is a special program that downloads "random" type files, any size.

RESTORE.CMD a special program to restructure copied "random" files for copying, or recopying back to the host system.

FREELINK.CMD a "bonus" utility that "relinks" the free chain of floppy or hard disk thereby eliminating fragmentation.

Completely documented source files included.

ALL 4 Programs (8" or 5") $99.50

Southeast Media

CHESS 6809

Requires FLEX and DISPLAYS On Any Type Terminal
Features:

*Four levels of play.
*Swap side. *Point scoring system.
*Two display boards. *Change skill level.
*Solve Checkmate problems in 1-2-3-4 moves.
*Make move and swap sides. *Play white or black.

This is one of the strongest CHESS programs running on any microcomputer, estimated USCF Rating 1600+ (better than most 'club' players at higher levels).

F and CCF - $79.95

VISA MasterCard

*FLEX is a trademark of Technical Systems Consultants
**OS9 is a trademark of Microware

TOLL FREE
1-800-338-6800
SOUTH EAST MEDIA
5900 Cassandra Smith Rd.
Hixson, TN 37343
info (615) 842-4801
CoCo OS-9™ FLEX™
SOFTWARE

Availability Legends —
F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

Southeast Media

DIET-TRAC Forecaster

DIET-TRAC Forecaster is an XBASIC program that plans a diet in terms of either calories and percentage of carbohydrates, proteins and fats (C P G%) or grams of Carbohydrate. Protein and Fat food exchanges of each of the six basic food groups (vegetable, bread, meat, skim milk, fruit and fat) for a specific individual.

Sex, Age, Height, Present Weight, Frame Size, Activity Level and Basal Metabolic Rate for normal individual are taken into account. Ideal weight and sustaining calories for any weight of the above individual are calculated. When a weight goal is given (either gain or loss), and a calorie plan is agreed upon between the computer and the individual, the number of days to reach the weight goal is projected. The starting and ending rate of weight loss is calculated, and a daily calendar with each day's weight for a 30-day period is printed.

F - $59.95
U - $89.95

Southeast Media

XDATA
A COMMUNICATION Package
for the UniFLEX Operating System

Allows UniFLEX Based Systems to Transmit and Receive files to and from other Computer Systems via Modem. Use with CP/M, Main Frames, other UniFLEX Systems, etc.

-- Verifies Transmission Integrity using checksum or CRC
-- Automatically Re-Transmits bad blocks
-- Transmits data in 128 byte blocks

U - $299.99

Southeast Media

JUS
Text Formatter

JUST, a Text Formatter developed by Ron Anderson, provides numerous features which make it a valuable addition to any FLEX Users Software Library. JUST is designed for formatting Text Output for Dot Matrix Printers and provides many unique features:

-Output the "Formatted" Text to the Display for format analysis and change.
-Output the "Formatted" Text to a Text File for use with the supplied FPRINT.CMD for producing multiple copies of the Text on the Printer INCLUDING EMBEDDED PRINTER COMMANDS (this Utility is very useful at other times also, and worth the price of the program by itself!).
-"User Configurable" for adapting to other Printers (comes set up for Epson MX-80 with Graftrax); provides for up to ten (10) imbedded "Printer Control Commands", such as Italics on and off, boldface on and off, etc.
-Automatic compensation for a "Double Width" printed line.
-Includes the normal line width, margin, indent, paragraph, space, vertical skip lines, page length, page numbering, centering, fill, justification, etc.
-Use with ANY Editor.
-Supplied with "Structured Source" (Windrush PL/9); easy to see the flow of the program.

F and CCF - $49.95

Lucidata

PASCAL UTILITIES
Requires LUCIDATA Pascal ver 3.

XREF -- produce a Cross Reference Listing of any text; oriented to Pascal Source.

F and CCF - $25.00

INCLUDE -- allows the inclusion of other Files in a Source Text; has unlimited nesting capabilities. Also allows Binary File inclusions.

F and CCF - $25.00

PROFILER -- produces an Indented, Numbered, "Structogram" of a Pascal Source Text File. Allows viewing the overall structure of large programs, and provides clues as to the integrity of the program. Supplied as Source Code; requires compilation.

F and CCF - $25.00

Lucidata

COPYCAT
Pascal MOT required

Allows reading TSC Mini-FLEX, SSB DOS68, and Digital Research CP/M Disks while operating under FLEX 1.0, FLEX 2.0, or FLEX 9.0 with 6800 or 6809 Systems. COPYCAT will not perform Miracles, but, between the program and the manual, you stand a good chance of accomplishing a transfer. Includes Utilities to List Directories, Copy Files, and convert Text Files when required. Also includes a Utility for investigating Physical Compatibility problems. Programs supplied in Modular Source Code (Assembly Language) to make it easier to solve unusual problems.

F and CCF 5" - $50.00
F 8" - $65.00

Computer Systems Consultants

FLEX DISK UTILITIES

Eight (8) different FLEX Utilities that should be a part of every FLEX Users Toolbox; Assembly Language (Source Code):

Copy a File with CRC Errors, so it can possibly be salvaged; Test Disk for errors; Compare two Disks; a fast Disk Backup Program; Edit Disk Sectors; Linearize Free-Chain on the Disk; print Disk Identification; and Sort and Replace the Disk Directory (in sorted order).

F and CCF - $50.00

WORD PROCESSORS

Alford and Associates

SCRIBBLER III

EXTREMELY Powerful Screen-Oriented Editor/Word Processor. Almost 50 different commands; EXCELLENT Documentation (over 300 pages), including a full Tutorial Section to help you learn how to use the system. Features Cursor-based editing, dynamic Screen Formatting (what you see is what you get), Multi-Column display and editing, "decimal align" columns (AND add them up automatically, if wanted), define multiple keystroke macros, even and odd page number headers and footers, imbed printer control codes in text, full justification series of commands, full "help" support, store common command series on disk for future use, etc. Easy "Set-Up" (for example, you just hit the key you want to use for a specific function, such as "cursor up", and the System reads an stores that key - no digging into tech manuals for codes, etc.); use supplied "set-ups", or remap the keyboard to what you are used too. Except for proportional printing, this package will DO IT ALL!

6800 or 6809 FLEX or SSB DOS, OS-9 - $175.00

Great Plains Computer Co.

STYLOGRAPH

A full-screen oriented WORD PROCESSOR -- (now runs on the Data-Comp and FHL Color FLEX Systems; uses the 51 x 24 Display Screens). Full screen display and editing (i.e., what you see is what you get); supports the Daisy Wheel proportional printers.

SPECIAL CCF - $195.00
F and O - $295.00 U - $395.00

SPELL

Fast Computer Dictionary.
F, CCF, OS/9 - $125.00 U - $175.00

MAIL MERGE

Greatly extends the power and flexibility of STYLOGRAPH.
F, CCF, O - $145.00 U - $195.00

VISA MasterCard

*FLEX is a trademark of Technical Systems Consultants
**OS9 is a trademark of Microware

TOLL FREE
1-800-338-6800
SOUTH EAST MEDIA
5900 Cassandra Smith Rd.
Hixson, TN 37343
Info (615) 842-4601
CoCo OS-9™ FLEX™
SOFTWARE

Availability Legends —
F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

Great Plains Computer Co.

MAIL MERGE

Greatly extends the power and flexibility of STYLOGRAPH. Allows Multiple Text files to be printed out as one large document. Provides for merging information into the Text File during printing (such as different names and addresses), etc.

F, CCF, O - $145.00
U - $195.00

Southeast Media

SPELL8 "Computer Dictionary"
OVER 120,000 words!

No more "Let your fingers do the walking through the Dictionary" while you are entering Text with your favorite Editor or Word Processor. **SPELL8** is more than just "another Spelling Checker"; it allows you to **look up a word** from within your Editor or Word Processor so that you **KNOW** it is right WHEN YOU TYPE IT IN with the SPN.CMD Utility (which operates in the FLEX Utility Space). Yes, it ALSO allows you to check and update the Text after you are finished; along with allowing you to ADD WORDS to the Dictionary, "Flag" questionable words in the Text for evaluation later, "View a word in context" before changing or ignoring, etc. SPELL8 first checks a "Common Word Dictionary", then the normal Dictionary, then a "Personal Word List", and finally, any "Special Word List" you may have specified. SPELL8 also allows the use of Small Disk Storage systems.

F and CCF - $129.95

Great Plains Computer Co.

SPELL

Fast Computer Dictionary -- allows directly changing the Text File, adding words to the dictionary, etc. 75,000 words in less than 400 sectors.

F, CCF, OS/9 - $125.00
U - $175.00

DATA BASE MANAGEMENT SYSTEMS

Southwestern Applied Business Systems

XDMS

Possibly one of the most powerful Database Management Systems' available, this machine language program is small enough to operate on a **single sided 5" disk**, yet provides the **speed** of M.L. and **power** limited only by the user's imagination. This DMS supports Relational, Sequential, Hierarchical, and Random Access File Structures, and has Virtual Memory capabilities for those Giant Data Bases. **XDMS Level I** provides a Functional "entry level" System which provides for defining a Data Base, entering and changing the Data, and producing Reports. **XDMS Level II** adds the POWERFUL "**GENERATE**" facility which uses an English Language Command Structure in manipulating the Data to create new File Structures, Sort, Select, Calculate, etc. **XDMS Level III** adds several special "Utilities" which provide additional ease of working with the various structures, changing System Parameters, etc.

XDMS Lvl I - F & CCF - $129.95
XDMS Lvl II - F & CCF - $199.95
XDMS Lvl III - F & CCF - $269.95
XDMS System Manual only - $24.95

Great Plains Computer Co.

DBMS/XBASIC DBMS

An XBASIC, Menu Driven, DBMS with "Built-In" Audit Tracking, Extremely Powerful Report & Format Capabilities, etc. This **Time** Proven DBMS will become the "Work Horse" of your Software Stable.

F and CCF $295.00
U $395.00

ACCOUNTING PACKAGES

Great Plains Computer Co. and **Universal Data Research, Inc.** both have Business Packages written in TSC xBASIC for FLEX, **CoCo FLEX**, and **UniFLEX** ----

• • • • • • • • • • • • •

- - - - Call **800-338-6800** for more information - - - -

• • • • • • • • • • • • •

Computer Systems Consultants

BASIC UTILITY PROGRAMS

Ten BASIC Programs to:

A BASIC Resequencer with EXTRAs over "RENUM"; works with ALL Versions of FLEX BASIC **AND** the Precompiler, checks for missing label definitions, processes Disk to Disk instead of in Memory.

Compare, Merge, or **Generate Updates** between two BASIC Programs, check BASIC Sequence Numbers, compare two unsequenced files, and 5 Programs for establishing a Master Directory of several Disks, and sorting, selecting, updating, and printing paginated listings of these files.

A BASIC Cross-Reference Program, written in Assembly Language, which provides an X-Ref Listing of the Variables and Reserved Words in TSC BASIC, XBASIC, and PRECOMPILER BASIC Programs.

ALL Utilities include Source (either BASIC or Source Code). An EXCELLENT Value!

F and CCF - $25.00
UniFLEX - $50.00

Computer Systems Consultants

FULL SCREEN INVENTORY/MRP

The Full Screen Inventory System provides a means of maintaining small inventories. Using a linked, keyed random file structure based upon the item field, it keeps the file in alphabetical order for easier inquiry. With the FIND command, the user may locate and/or print all records matching on partial or complete item, description, vendor, or attributes. Items in backorder or below minimum stock levels may be located and/or printed thru the same process. Printed output may be produced in item or vendor order. A materials requirement planning (MRP) capability for manufacturing environments is included to allow the maintenance and analysis of Hierarchical assemblies of items in the inventory file. It requires TSC's Extended BASIC.

F and CCF - $100.00. U - $150.00

ELECTRONIC SPREADSHEET

The Virginia Company ***Bizpack***

BIZPACK is used for storing accounting, numeric, and financial data which can then be used for planning, budgeting, forecasting, analyzing, etc. While "Electronic Spreadsheets" are extremely useful in many situations, BIZPACK excels in businesses where there are numerous expense columns, revenue sources, significant business indicators, large numbers, erratic week-to-week and month-to-month fluctuations, etc. BIZPACK helps determine statistical relationships, establish trend lines, "smooths" data via moving averages, analyze seasonal data, adjusts for inflation, lags data in Statistics or Column functions, plots data, etc. BIZPACK is oriented toward time series analysis of businesses. The Program displays information on the screen in Columns of Information with each Row conforming to a defined Period of Time (weeks, months, years, etc.), and is very easy to use (data is easy to enter, change, and modify; commands can be renamed to suit the users requirements; unlimited ability to create specialized commands using common BASIC Statements; etc.). Requires TSC's Extended BASIC.

F and CCF - $135.00
with Source - $250.00

*** SPECIAL ***

Purchase XBASIC and BIZPACK together for $221.50
-- a Savings of $13.50 --

VISA MasterCard

*FLEX is a trademark of Technical Systems Consultants
*OS9 is a trademark of Microware

Availability Legends —
F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

*** SPECIAL ***
Purchase XBASIC and EDITPACK together for $221.50
— a Savings of $13.50 —

Computer Systems Consultants
TABULA RASA SPREADSHEET
TABULA RASA is similar to DESKTOP/PLAN and provides for the generation and maintenance of tabular computation schemes often used for analysis of business, sales, and economic scenarios. Its menu-driven user interface provides these capabilities even to those users with no programming experience. Its extensive report-generation capabilities allow the user to generate professional results with minimum effort. It requires TSC's Extended BASIC.
F and CCF - $100.00. U - $125.00

Computer Systems Center
DYNACALC
THE Electronic Spread Sheet for 6809 Computer Systems. An extremely POWERFUL Business Tool, this Program will find an unlimited number of "non-business" applications, also (for example, a full Junior College Electronics Curriculum was set up using DYNACALC). Advanced features like "Table Lookup" make Income Tax work easy; Column or Row Sorting for numerous applications; etc. Completely "Memory Resident", Machine Language, this Program is FAST. Provides STANDARD FLEX Text File output for use with BASIC, Word Processors, Pascal, "C", etc. Also available for Data-Comp and FHL FLEX systems using the 50 x 24 Displays.
F and SPECIAL CCF - $200.00
U - $395.00

ODDS & ENDS

Computer Systems Consultants
FULL SCREEN FORMS DISPLAY
This package supports any Serial Terminal with cursor control of Memory-Mapped Video Displays. The package substantially extends the screen Input/Output capabilities of TSC's Extended BASIC programs by providing a simple, table-driven method of describing and using full screen displays. These table entries are easy to set up and maintain, and are normally stored on disk and read as required. A simple, interactive means of generating the forms and the data field definitions is provided.
F and CCF - $50.00, U - $75.00

Computer Systems Consultants
FULL SCREEN MAILING LIST
The Full Screen Mailing List System provides a means of maintaining simple mailing lists. Using a random fill structure based on the first character of the name field, it maintains the file in alphabetical order for easier inquiry. With the FIND command, the user may locate all records matching on partial or complete name, city, state, zip, or attributes. Printed listings and output to labels may also be produced on the same selective basis. It requires TSC's Extended BASIC.
F and CCF - $100.00, U - $110.00

COLOR COMPUTER SOFTWARE

Stearns Electronics
FORTH
Intrigued by Forth?? Here is a FORTH package tailored to the Color Computer! This package is supplied on Tape, with instructions for transferring it to disk if you wish. Written primarily in machine language, it's speed is unparalleled. A full Semigraphic-6 Editor is provided, along with "goodies" like Graphics and Sound Commands, Printer Commands, Auto-Repeat and Control Keys, etc. If you are interested in learning FORTH, a Trace Feature is provided which is invaluable. If you are a FORTH Pro, this package provides CPU carry Flag accessability, Fast Task Multiplexing, Clean Interrupt Handling, etc. (Or: you won't "out grow" the Basic capabilities of this Implementation). Combine this package with Leo Brodie's EXCELLENT book "Starting FORTH", and you will be a FORTH Expert before you know it (and have a lot of fun doing it!).
Color Computer TAPE - $98.95

Custom Software Engineering, Inc.
Color Computer GRAPHIC SCREEN PRINT Programs
Dumps any "PMODE" Screen to the Printer with the BASIC USR Function. Shift the Printout Left or Right or Reverse Print (Dark for Light Screen and Vice Versa). All Programs on Tape.

CSPR for R.S. LP-VII/VIII & DMP 100/200/400	$7.95
CSPE for Epson w/ Graftrax and Graftrax +	$9.95
CSPG for Gemini 10 and 15	$9.95
CSPP for the Prowriter Printers	$9.95

Custom Software Engineering, Inc.
DATE-O-BASE CALENDAR Program
A Menu Driven EXTENDED BASIC Program which allows the entry of up to 12 Memos per Day, each of which may contain up to 28 Characters, for any day of the Month between the years 1700 and 2099. A Graphic Calendar shows which days contain Memos, and a "Key Word" Search is provided which can be output to the Screen or Printer.

TAPE DATE-O-BASE CALENDAR
(Each Tape File will hold up to 400 Memos) $16.95
DISK DATE-O-BASE CALENDAR
(4,000 Memos at 300/Month per Disk) $19.95

Custom Software Engineering, Inc.
That's INTEREST-ing
Interested in INTEREST (the Money Kind)? An EXTENDED BASIC Program that will help you deal with numerous problems requiring interest calculations. Present Value, Rate of Return, Current Bond Yield and Rate of Return to maturity, Loan Repayment Amortization Schedules, etc.
TAPE - $29.95

Custom Software Engineering, Inc.
DISK DATA HANDLER 64K
An EXTENDED BASIC Data Management System w/ Mach. Lang. Routines. Allows a max of 246 Chars. and 14 Fields per Record, and another Record can be linked to the first: 8 Char. Field Names, up to 99 Chars. per Field. Powerful On-Screen editor for input and update, flexible Output capabilities including output to Disk Files for use by other Programs. Change File Definition without re-entering the Data, Split Files, etc. Allows Multiple Field Sorts, Select on any combination of Fields, etc. An extremely POWERFUL TOOL; instructions provide examples of Mailing Lists and a Financial Stock Profit and Loss Tracking System.
DISK - $54.95

Custom Software Engineering, Inc.
DISK DOUBLE ENTRY
DISK EXTENDED BASIC Accounting Program w/ Mach. Lang. Routines. A "Traditional" Accounting Package for Small Businesses, Clubs, Churches, Personal Use, etc. Up to four levels of subtotals with Trial Balance, Income Statement, and Balance Sheet Reports. Not allows up to 300 accounts and a Trial Balance of $9,999,999.99. Transactions may be up to 14 lines long, and comments and explanations may be freely used. Accounts are traceable to the journal transaction, which may include comments. Screen reports allow review of past transactions and current balances.
DISK - $44.95

Availability Legends —
F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

Computer Systems Center

DYNASHARE

— Multi-User, Multi-Tasking with **FLEX** —

Southeast Media is now shipping DYNASHARE **FROM STOCK** - the multi-user, multi-tasking capability of **DYNASHARE** allows FLEX users the advantages of more sophisticated and time saving computer usage without having to buy or learn a new Language or Operating System syntax. **DYNASHARE**, as its name implies, allows true "time-sharing" operation under the popular FLEX Operating system, and also allows each user to run two simultaneous jobs (multi-tasking); even on single-user systems. For example, while in EDIT, you can list another file or examine a directory. Or, you might look up an item in a Data Base while a Sort is in progress! **DYNASHARE** also provides some fringe benefits that will be greatly appreciated by FLEX users, including type-ahead, command line editing, and instant response to "escape".

DYNASHARE is the painless method! Use your existing Flex computer by simply adding 64K of RAM for each user and/or task. Fact is, you still use FLEX just like you always have! **DYNASHARE** is not intended as competition to UniFLEX. It does not improve on the speed of FLEX, and does not offer password protection or other niceties of a full-blown multi-user system. What **DYNASHARE does do** is give FLEX users a **low-cost way to use existing software in a multi-user, multi-tasking environment**, so your existing FLEX versions of BASIC, XBASIC, editors, assemblers, disassemblers, sort/merge packages, word processors, compilers, DYNACALC spread-sheet package, and so on are still good.

NOTE -- The **initial release** of DYNASHARE is for **6809** Computers, but versions will also be available for other popular extended-memory (up to **1024K**) systems, such as HELIX and CMIX. A minimum of **128K** of **RAM** will be required with **ALL** versions. **DYNASHARE** requires **64K of RAM for each active task**: thus a 256k system could allow foreground-background operation on two terminals, or foreground-only operation on four terminals.

AVAILABLE NOW from **Southeast Media** - **$200.00**

AUTHORS - PROGRAMMERS
QUALITY SOFTWARE NEEDED
FLEX - UniFLEX - **OS/9** - **Color Computer**

For the past several months, we at the **Southeast Media Division of Computer Publishing, Inc.** (CPI), the parent company of **'68' MICRO JOURNAL** and **COLOR MICRO JOURNAL**, have debated expanding our software distribution business. Many other magazines have been doing so for years (in fact, MOST were in the Software Distribution Business BEFORE they began to publish a Magazine). Presently there are many fine examples of software that has been developed by YOU, our readers, that will never see the "light of day" due to the **Cost of Advertising** and **TIME and Cost** involved in the **production, distribution,** and **Customer SUPPORT** of that software unless SOMEONE, with enough exposure and the willingness to continually advertise, runs with the ball.

Software is the "backbone" for the REAL utilization of any Computer System, and ours are no exception! This has been no simple decision. While we realize that there could be some conflict with some of our advertisers, we ALSO hear a LOUD and CONTINOUS cry for **HELP** from our Readers. From day one, the **foremost concern** of '68' MICRO JOURNAL has been it's **READERS!** Therefore, our **Southeast Media Division** will accept, for appraisal for possible Distribution, 6809 software: Games, Utilities, Software Development, Business Application Programs, etc.

In the past there has been **too much** software offered that was not quite ready. We will strive to eliminate that element. But, right up front, we tell you only that we will do our very best; nothing more. Also, we will strive to keep cost to a bare minimum, while securing for the author a fair return in royalty payments, promptly paid, and in customer support for his product.

Of course, we will expect, no -- **DEMAND**, that the author keep the product free of errors (bugs), and maintain it in a prompt and business like manner. Also we shall require that authors be willing to furnish 'source' for those programs that justify, by price and utility, inclusion of same. The lack of source code, properly commented, is a continual complaint we hear. Not all programs will be sold with source, but where necessary, we will insist that it be included.

In some instances the program may be small or short and not justify itself as a "single" sale product. In this event it will be combined with other like programs, and offered as a package. In that event, the royalties will be split between the various authors.

If you have software that you feel will qualify under this program, please contact one of the people below. Remember, if your software has any problems or "funnies" -- **GET IT STRAIGHT BEFORE YOU CONTACT US!!** Also get your source code in proper shape and well commented; there is too much 99% code already drifting around.

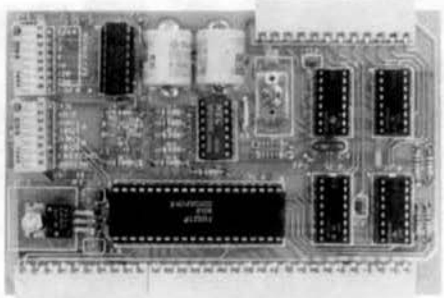
If your software is **READY** contact:
Bob Ray, Don Williams, or Tom Williams

Southeast Media is a division
of **Computer Publishing, Inc. (CPI)**,
a family of 100% 68XX support facilities.

*FLEX is a trademark of Technical Systems Consultants
*OS9 is a trademark of Microware

Availability Legends —
F = FLEX, **CCF** = Color Computer FLEX
O = OS-9, **CCO** = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

CALENDAR-CLOCK / TIMER / PARALLEL PORT



Calendar - Clock CLK68-1

- [illegible]
- All clock functions Software controlled
- [illegible]
- [illegible]

Interval Timer

- [illegible]
- [illegible]
- OS-9 runs faster with CLK68-1 than with timers such as SWTPC MP-T
- [illegible]

Parallel I/O Port -- [illegible]

- [illegible]
- [illegible]

Construction -- [illegible]

Manual -- Well documented - 36 pages

Dealer & OEM discount available

Assembled and tested	$119.95	Kit	$89.95
Goldplated bus conn	7.50	2 MHz option	2.50
Disk 5 or 8 in. SSB or Flex* OS-9 Available NOW			14.95

** [illegible]
* [illegible]

ROBERTSON ELECTRONICS
1003 Warm Sands Dr. SE
Albuquerque, NM 87123

Phone (505) 294-0025
NM residents add 4% tax
Add $3 Shipping & Handling

COMPARE

our EPROM PROGRAMMER with the field.

All data taken directly from manufacturer's current advertising. Software, interfaces, or personality modules may also be required at additional cost.

- • Triple voltage EPROM
- • Supplied in kit form

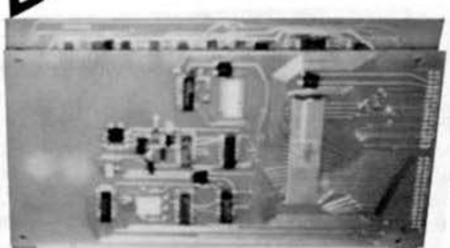
		A	B	C	D	E	F
INTERFACE	S30	PAR	PAR	SER	S30	SER	SER
INTELLIGENT	NO	NO	NO	YES	NO	YES	YES
PROGRAMS							
2704•			●				●
2508	●			●	●	●	
2708•			●		●		●
275	●	●	●	●		●	●
2516	●	●	●	●	●	●	
2716	●	●	●	●	●	●	●
2716•			●		●		●
2532	●		●	●	●	●	●
2732	●		●	●	●	●	●
2732A	●		●		●	●	●
2564	●		●		●	●	●
2764	●		●		●	●	●
2526	●				●		
27128	●						
2816							●
68764			●				
6748						●	
6749						●	
TOTAL	11	3	12	6	11	11	11
PRICE	$125	$45•	$169	$289	$375	$489	$575

EPRAM EPROM Programmer, $125. Personality module for 2508, 2758, 2516, and 2716 included. Specify CPU, disk size, and operating system (TSC's FLEX or SSB's DOS) when ordering. Manual only, $10; refundable with EPRAM purchase.

UNITEK • P.O. Box 671 • Emporia, VA 23847

SUPER CPU ... only from LSI

68008



449.95 KIT

We're mighty proud of our new processor card. We're giving you the ability to go 68000 without major changes to your system. Our new CPU gives you these advanced features:

- Dynamic partitioning memory management unit with bound check register.
- On-board timer for multi-user/multi-tasking applications.
- On-board boot-strap EPROM and Monitor EPROM space.
- Vectored priority interrupt generator.
- On-board wait state generator.
- User selectable bus options that includes a new higher bandwidth bus mode.
- And many more . . .

S68K/08-CPU- SSEM. & TESTED 549.95
S68K/08-CPU-KIT ORM 449.95
(KIT INCLUDES PROCESSOR, CRYSTAL, SOCKETS AND CONNECTORS)
DISK CONTROLLER SUPPORTED DC3, DC4, DMF2, SDC8.

Announcing . . .

THE *SHELL* FOR FLEX 9™ *

We are pleased to announce the *SHELL*, a UNIX ++ like shell that supports I/O redirection, pipes, macro substitution and programmable shell scripts! The shell will work with all your existing programs and utilities. Requires 56K of user ram, FLEX 9™ * version 2.6 and above. The shell occupies the top 8K of user ram. An excellent tool for the 6809 community.

FLX/SHO9-8 . . 8 inch version 90.00
FLX/SHO9-5 . . 5.25 inch version . . . 90.00
ONE YEAR MAINTENANCE 22.50

LSI STANDS BEHIND ITS PRODUCTS
1 YEAR LIMITED WARRANTY
ON ALL OUR PRODUCTS

ANNOUNCING THE LSI 68000 USERS GROUP

Join the Group by sending us your name and address. You will receive our monthly publication with free public domain user programs and software updates.
New members of the users group will receive a $30.00 discount on THEIR FIRST LSI hardware purchase. Anyone that donates a program to the group will receive our current user group software on a formatted CP/M readable disk.

ANY ORDERS RECEIVED ON SDC8 OR THE 256K RAM CARD BEFORE 4-1-84 WILL RECEIVE A $50 DISCOUNT

256K RAM CARD
Using the latest LSI technology this 256K RAM CARD makes a perfect addition to your S68K system. Uses MRDY for refresh arbitration.
S68K/08-256K . . . $780.00

SDC8 CONTROLLER
A SS50 DMA disk controller for use with either 68000 bus modes or 6809 bus modes. Features a high reliability digital data separator. (No analog circuits to drift) and full 1 Megabyte addressing range.
S68K/08-SDC8-$550.00

CP/M-68K +
DRI's famous CP/M made for the 68000. It includes a C compiler, relocatable assembler, linking loader, librarian and many utilities. It is source compatible with all other CP/M operating systems.
CP/M-68K + . . . $350.00

N.Y. residents add sales tax.

VISA master charge

and COD accepted

*FLEX 9™ is a trademark of Technical Systems Consultants Inc.
+ CP/M-68K is a registered trademark of Digital Research, Inc.
++ UNIX is a registered trademark of Bell Labs.
All prices and offers subject to change without notice.

LSI Enterprises Ltd.
PO Box 1227
Woodhaven, NY 11421
(212) 423-5596

TEN MOST-ASKED QUESTIONS about DYNACALC™

THE ELECTRONIC SPREAD-SHEET FOR 6809 COMPUTERS

1. What is an electronic spread-sheet, anyway?
Business people use spread-sheets to organize columns and rows of figures. DYNACALC simulates the operation of a spread-sheet without the mess of paper and pencil. Of course, corrections and changes are a snap. Changing any entered value causes the whole spread-sheet to be re-calculated based on the new constants. This means that you can play, 'what if?' to your heart's content.

2. Is DYNACALC just for accountants, then?
Not at all. DYNACALC can be used for just about any type of job. Not only numbers, but alphanumeric messages can be handled. Engineers and other technical users will love DYNACALC's sixteen-digit math and built-in scientific functions. You can build worksheets as large as 256 columns or 256 rows. There's even a built-in sort command, so you can use DYNACALC to manage small data bases — up to 256 records.

3. What will DYNACALC do for ME?
That's a good question. Basically the answer is that DYNACALC will let your computer do just about anything you can imagine. Ask your friends who have VisiCalc™, or a similar program, just how useful an electronic spread-sheet program can be for all types of household, business, engineering, and scientific applications. Typical uses include financial planning and budgeting, sales records, bills of material, depreciation schedules, student grade records, job costing, income tax preparation, checkbook balancing, parts inventories, and payroll. But there is no limit to what YOU can do with DYNACALC.

4. Do I have to learn computer programming?
NO! DYNACALC is designed to be used by non-programmers, but even a Ph.D. in Computer Science can understand it. Even experienced programmers can get jobs done many times faster with DYNACALC, compared to conventional programming. Built-in HELP messages are provided for quick reference to operating instructions.

5. Do I have to modify my system to use DYNACALC?
Nope. DYNACALC uses any standard 6809 configuration, so you don't have to spend money on another CPU board or waste time learning another operating system.

6. Will DYNACALC read my existing data files?
You bet! DYNACALC has a beautifully simple method of reading and writing data files, so you can communicate both ways with other programs on your system, such as the Text Editor, Text Processor, Sort/Merge, STYLOGRAPH™ word processor, RMS™ data base system, or other programs written in BASIC, C, PASCAL, FORTRAN, and so on.

7. How fast is DYNACALC?
Very. Except for a few seldom-used commands, DYNACALC is memory-resident, so there is little disk I/O to slow things down. The whole data array (worksheet) is in memory, so access to any point is instantaneous. DYNACALC is 100% 6809 machine code for blistering speed.

8. Is there a version of DYNACALC for MY system?
Probably. You need a 6809 computer (32k minimum) with FLEX™, UniFLEX™, or OS-9™ operating system. You also need a decent crt terminal, one with at least 80 characters per line, and direct cursor addressing. If your terminal isn't smart enough for DYNACALC, you probably need a new one anyway. The UniFLEX and OS-9 versions of DYNACALC allow you to mix different brands of terminal on the same system. There's also a special version of DYNACALC for Color Computers equipped with FLEX (Frank Hogg or Data-Comp versions).

9. How much does DYNACALC cost?
The FLEX versions are just $200 per copy; UniFLEX version $395; OS-9 version (works with LEVEL ONE or LEVEL TWO) $250. Orders outside North America add $7 per copy for postage. We encourage dealers to handle DYNACALC, since it's a product that sells instantly upon demonstration. Call or write on your company letterhead for more information.

10. Where do I order DYNACALC?
See your local DYNACALC dealer, or order directly from CSC at the address below. We accept telephone orders from 10 am to 6 pm, Monday through Friday. Call us at 314-576-5020. Your VISA or MasterCard is welcome. Please specify diskette size for FLEX or OS-9 versions. Software serial number is required for the UniFLEX version.

Order your DYNACALC today!

Foreign Dealers:

Australia & Southeast Asia: order from Paris Radio Electronics, 161 Bunnerong Road (PO Box 380) Kingsford, 2032 NSW Australia. Telephone: 02-344-9111.

United Kingdom: order from Compusense, Ltd., PO Box 169, London N13 4HT. Telephone: 01-882-0681.

Scandinavia: order from Swedish Electronics hk AB, Murargatan 23-25, Uppsala S-754 37 Sweden. Telephone: 18-25-30-00.

Computer Systems Center
13461 Olive Blvd.
Chesterfield, MO 63017
(314) 576-5020

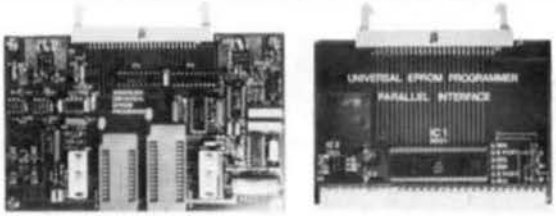
UniFLEX software prices include maintenance for the first year.

DYNACALC is a trademark of Computer Systems Center

VisiCalc is a trademark of VisiCorp.
STYLOGRAPH is a trademark of Great Plains Computer Co.
RMS is a trademark of Washington Computer Services.
FLEX and UniFLEX are trademarks of TSC.
OS-9 is a trademark of Microwa e and Motorola.

WINDRUSH MICRO SYSTEMS

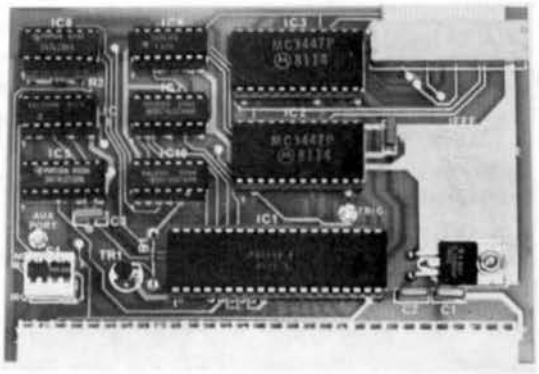
UNIVERSAL EPROM PROGRAMMER



- PROGRAMS and VERIFIES 2508, 2708, 2516, 2716, 2532, 2732A, 2564, and 2764 EPROMS. Minor hardware mods are required to program the INTEL 27128.
- Tri-volt and Single Volt 2508/2708 and 2516/2716 devices are supported.
- ZIF sockets with mode selector switches eliminate 'personality modules'.
- Twin boards with five feet of twisted pair planar cable puts the programmer out on the bench where it belongs.
- SS-30 and EXORCISOR interfaces are available.
- Menu driven software provides the following facilities:
  a. MOVE blocks of memory within the buffer.
  b. READ an EPROM into the buffer.
  c. VERIFY an EPROM against the buffer.
  d. EXAMINE and change the contents of the buffer.
  e. DUMP the contents of the buffer in HEX and ASCII.
  f. FILL a selected area of the buffer with a specified character.
- Software available for all versions of SSB DOS, FLEX 2, FLEX 9 and OS-9. Assembly language source files supplied on disk....enables customising.
- Well documented users manual provides step-by-step adaptation and operating instructions.

AVAILABLE FROM GIMIX IN THE U.S.A.

IEEE-488



- SUPPORTS ALL PRINCIPAL MODES OF THE IEEE-488 (1975/81 BUS SPECIFICATION:
  - Talker
  - Listener
  - System Controller
  - Serial Poll
  - Parallel Poll
  - Group Trigger
  - Single or Dual Primary Address
  - Secondary Address
  - Talk only...Listen only
- Fully documented with a complete reprint of the KILOBAUD article on the IEEE bus.
- Low level assembly language drivers suitable for 6800, 6801, 6802, 6803, 6808 and 6809 are supplied in the form of listings. These drivers have been extensively tested and are GUARANTEED to work!
- Single SS-30 board (4, 8, or 16 addresses per port), fully socketed, gold plated bus connectors, and IEEE interface cable assembly.

PL/9

EDITOR/COMPILER/DE-BUGGER

- Friendly inter-active environment where you have INSTANT access to the Editor, the Compiler, and the Trace-Debugger, which, amongst other things, can single step the the program a SOURCE line at a time. You also have direct access to any FLEX utility and your System Monitor.
- 250 page manual is organized as a tutorial with plenty of examples.
- Fast single pass compiler produces 8K of COMPACT and FAST 6809 machine code output per minute with no run-time overheads or license fees.
- Fully compatible with TSC text editor format disk files
- Signed and unsigned BYTEs and INTEGERs, 32-bit floating point REALs.
- Vectors (single dimension arrays) and Pointers are supported.
- Mathematical expressions: (+), (-), (*), (/), modulus (\), negation (-)
- Expression evaluators: (=), (<>), (<), (>), (>=), (<=)
- Bit operators: (AND), (OR), (EOR/XOR), (NOT), (SHIFT), (SWAP)
- Logical operators: (.AND), (.OR), (.EOR/.XOR).
- Control statements: IF..THEN..ELSE, IF..CASE1..CASE2..ELSE, BEGIN..END, WHILE.., REPEAT..UNTIL, REPEAT..FOREVER, CALL, JUMP, RETURN, BREAK, GOTO.
- Direct access to (ACCA), (ACCB), (ACCD), (CCR) and (XREG).
- FULLY supports the MC6809 SWI, SWI2, SWI3, NMI, FIRQ, IRQ and RESET vectors. Writing a self-starting (from power-up) program that uses ANY, or ALL, of the MC6809 interrupts is an absolute snap!
- Procedures may be passed and may return variables. This makes them functions which behave as though they were an integral part of PL/9.
- Several fully documented library function modules are supplied: IOSUBS, BITIO, HAMDIO, HEXIO, FLEXIO, SCIPACK, STRSUBS, and REALCOM.

'... THIS IS THE MOST EFFICIENT COMPILER I HAVE FOUND TO DATE.'

Quoted from Ron Anderson's FLEX User Notes column. Need we say more?

MACE/XMACE

A co-resident EDITOR/ASSEMBLER for the 6809 written by Graham Trott which takes most of the pain out of assembly language program development:

- Friendly inter-active environment where you have INSTANT access to the Editor, the Assembler, FLEX and your System Monitor.
- MACE can also produce ASMPEDC's for PL/9 with the assembly language source passed to the output file as comments.
- Includes XMACE a co-resident 6800/1/2/3/8 EDITOR/CROSS ASSEMBLER.

'C'

This is the FLEX version of the James McCosh 'C' compiler that is also available on UNIFLEX from SWTP and OS-9 from Microware!

- The FLEX implementation supports the full Kernighan and Ritchie 'C' specification except 'floats', 'doubles', and 'bit-fields'.
- Produces very efficient assembly language source output with the 'C' source optionally interleaved as comments.
- Built-in optimizer will shorten object code by about 11%
- Supports interleaved assembly language programs.
- The TSC relocating assembler/linking loader (SPOO-17) is REQUIRED.

MACE	(6809 FLEX only)	$ 98.00
PL/9	(6809 FLEX only)...(a steal at this price!)	$198.00
'C'	(a 56K system and the TSC SPOO-17 package is req'd)	$295.00
IEEE-488	with IEEE-488 cable assembly	$298.00
SS-30	Universal EPROM programmer w/one version of software	$375.00
EXORciser	Universal EPROM programmer w/one version of software	$395.00
SOFTWARE	Drivers for a 2nd, 3rd, or 4th operating system	$ 25.00

ALL PRICES INCLUDE AIR MAIL POSTAGE

WE STOCK THE FOLLOWING COMPANIES PRODUCTS: GIMIX, SSB, FHL, MICROWARE, TSC, LUCIDATA, AND ALFORD & ASSOCIATES.

FLEX (tm) is a trademark of Technical Systems Consultants, OS-9 (tm) is a trademark of Microware Systems Corporations, MDOS (tm) and EXORciser (tm) are trademarks of Motorola Incorporated.

An SS-50C all CMOS 256K STATIC RAM board will be available SOON!

Write for details & pricing.

WORSTEAD LABORATORIES, NORTH WALSHAM, NORFOLK, ENGLAND. NR28 9SA.
TEL: (0692) 405189
TLX: 97380 SHARET G

FEATURES THE POWERFUL, THIRD GENERATION, MOTOROLA 6809 PROCESSOR!

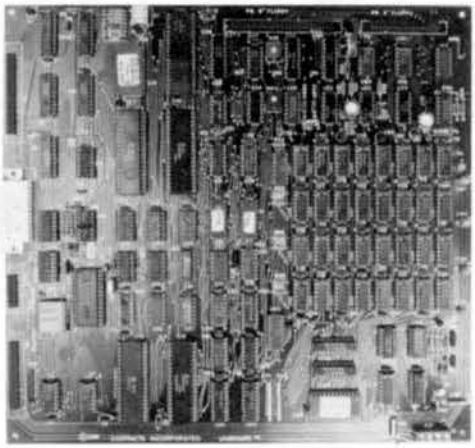
THE 6809 "UNIBOARD"™

SINGLE BOARD COMPUTER KIT

New!

PERFECT FOR COLLEGES, OEM'S, INDUSTRIAL AND SCIENTIFIC USES!

64K RAM! DOUBLE DENSITY FLOPPY DISK CONTROLLER!



BLANK PC BOARD
$99.95
WITH PAL'S, AND TWO EPROMS.

FOR 5-1/4 OR 8 INCH SOURCE DISKETTE ADD $10.

$399.00
COMPLETE KIT! FULLY SOCKETED.

ALL OPTIONS ARE STANDARD. NO EXTRAS TO BUY!

THE COMPACTA UNIBOARD™: Through special arrangement with COMPACTA INC., we are proud to have been selected the exclusive U.S. Mfg. of their new 6809 UNIBOARD™ COMPUTER KIT. Many software professionals feel that the 6809 features probably the most powerful instruction set available today on ANY 8 bit micro. Now, at last, all of that immense computing power is available at a truly unbelievably low price.

FEATURES:

* 64K RAM using 4116 RAMS.
* 6809E Motorola CPU.
* Double Density Floppy Disk Controller for either 5-1/4 or 8 inch drives. Uses WD1793.
* On board 80 x 24 video for a low cost console. Uses 2716 Char. Gen. Programmable Formats. Uses 6845 CRT Controller.
* ASCII keyboard parallel input interface. (6522)
* Serial I/O (6551) for RS232C or 20 MA loop.
* Centronics compatible parallel printer interface. (6522)
* Buss expansion Interface with DMA channel. (6844)
* Dual timer for real time clock application.
* Powerful on board system monitor (2732). Features commands such as Go To, Alter, Fill, Move, Display, or Test Memory. Also Read and Write Sectors. Boot Normal, Unknown, and General Flex™.

YOUR CHOICE OF POPULAR DISK OPERATING SYSTEMS:

FLEX™ from TSC	$149
OS9™ from Microware	$199
Specify 5-1/4 or 8 inch	

PC BOARD IS DOUBLE SIDED, PLATED THRU SOLDER MASKED, 11 x 11-1/2 IN.

ALL SALES ARE MADE SUBJECT TO THE TERMS OF OUR 90 DAY LIMITED WARRANTY. A FREE COPY IS AVAILABLE UPON REQUEST.

Digital Research Computers
(OF TEXAS)
P.O. BOX 461565 • GARLAND, TEXAS 75046 • (214) 271-3538

TERMS: Shipments will be made approximately 3 to 6 weeks after we receive your order. VISA, MC, cash accepted. Add $4.00 shipping. USA AND CANADA ONLY

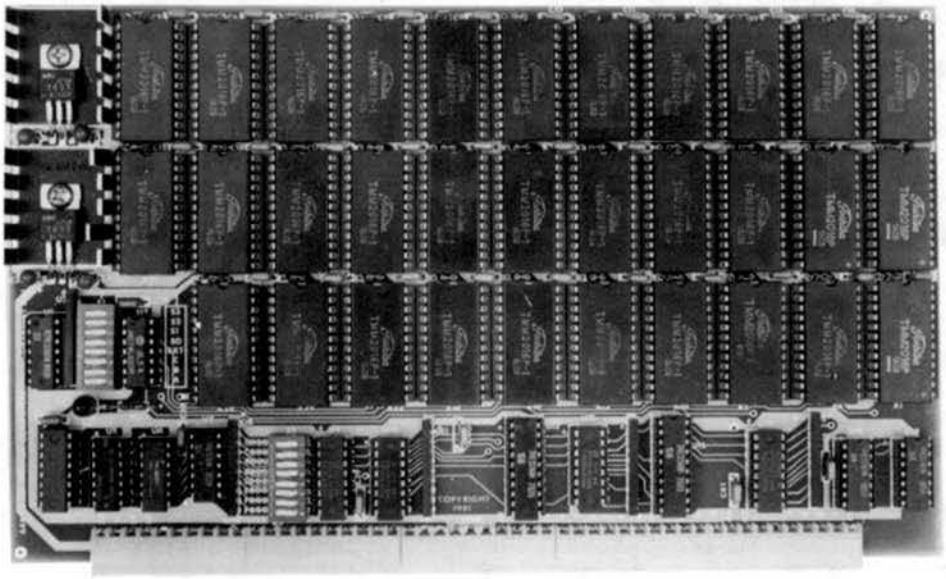
64K SS-50 STATIC RAM

NEW!

$179.00 (48K KIT)

NEW!

LOW POWER!



RAM OR EPROM!

BLANK PC BOARD WITH DOCUMENTATION $52

SUPPORT ICs + CAPS -	$18.00
FULL SOCKET SET -	$15.00

ASSEMBLED AND TESTED **ADD $50**

FEATURES:

- ★ Uses new 2K x 8 (TMM 2016 or HM 6116) RAMs.
- ★ Fully supports Extended Addressing.
- ★ 64K draws only approximately 500 MA.
- ★ 200 NS RAMs are standard. (TOSHIBA makes TMM 2016s as fast as 100 NS. FOR YOUR HIGH SPEED APPLICATIONS.)
- ★ Board is configured as 3-16K blocks and 8-2K blocks (within any 64K block) for maximum flexibility.
- ★ 2716 EPROMs may be installed anywhere on Board.
- ★ Top 16K may be disabled in 2K blocks to avoid any I/O conflicts.
- ★ One Board supports both RAM and EPROM.
- ★ RAM supports 2MHZ operation at no extra charge!
- ★ Board may be partially populated in 16K increments.

56K	$219
64K	$249

CLOSE OUT SPECIAL

WE HAVE DROPPED OUR 32K SS-50 STATIC RAM BOARD WHICH USED 2114 LOWPOWER RAMS. WE WILL SELL THE REMAINING STOCK OF BLANK PCB'S WITH DATA FOR $17.50 EA. THESE FORMERLY SOLD FOR $50.

16K STATIC RAMS?

The new 2K x 8, 24 PIN, static RAMs are the next generation of high density, high speed, low power, RAMs. Pioneered by such companies as HITACHI and TOSHIBA, and soon to be second sourced by most major U.S. manufacturers, these ultra low power parts, feature 2716 compatible pin out. Thus fully interchangeable ROM/RAM boards are at last a reality, and you get BLINDING speed and LOW power thrown in for virtually nothing.

Digital Research Computers
(OF TEXAS)
P.O. BOX 401565 • GARLAND, TEXAS 75040 • (214) 271-3538

TERMS: Add $2.00 postage. We pay balance. Order under $15 add 75¢ handling. No C.O.D. We accept Visa and MasterCharge. Tex. Res. add 5% Tax. Foreign orders (except Canada) add 20% P & H. Orders over $50, add 85¢ for insurance.

68' MICRO JOURNAL ADVERTISERS INDEX

This index is provided as a reader service. The publisher does not assume any liability for omissions or errors.

THOMAS INSTRUMENTATION

HARDWARE	ASSEM & TEST	BARE CARD
CPU-5	$299.00	$49.00
S-R/R w/o memory chips	$120.00	$49.00
S-R/R with 48K	$399.00	$49.00
SP-1	$195.00	$49.00
6802 SUPER CPU	$235.00	$59.00
VIDEO RAM	$195.00	$49.00
PARALLEL I/O	$139.00	$49.00
SS-50/50C EXTENDER	$ 35.00	NA
SS-30 EXTENDER	$ 25.00	NA
SS-50 WIRE WRAP	NA	$39.00
SS-30 WIRE WRAP	NA	$20.00
SS-30 BACKPLANE 8POS.	NA	$39.00
SS-50 BACKPLANE 4,8,12, & 16 @ $5.00/SLOT	NA	NA

SOFTWARE

VDISK
Use extended memory as a fast disk drive
6809 source & obj $149.00
6809 object $ 99.00

OUTSIDE MODEM PROGRAM
includes source
UniFLEX™ $100.00
FLEX™ 6800 or 6809 $ 50.00

CROSS ASSEMBLER
For 6800, 6801, 6805
Runs on 6809 FLEX™
$150.00

COMPONENTS

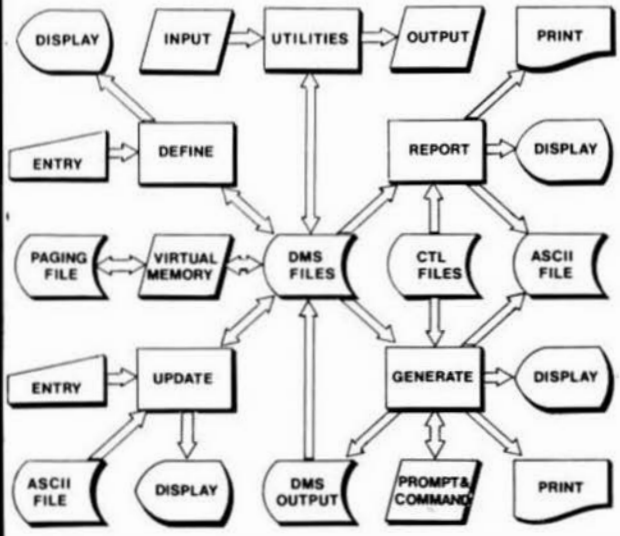
GOLD MALE MOLEX $1.60 GOLD FEMALE MOLEX $1.60
TIN MALE MOLEX $ .40 TIN FEMALE MOLEX $ .50
146805E2P $20.00

MASTERCARD, VISA, AND C.O.D. ACCEPTED. PLEASE ALLOW 3 WEEKS FOR CHECKS TO CLEAR. CONT. USA ADD $3.00 SHIPPING. CANADA $6.00. FOREIGN $12.00. NJ RESIDENTS ADD 6% SALES TAX
FLEX and UniFLEX ARE TRADEMARKS OF TECHNICAL SYSTEMS CONSULTANTS

THOMAS INSTRUMENTATION
168 EIGHTH STREET
AVALON, N.J. 08202
(609) 967-4280

XDMS

Data Management System



System Architecture

WEBTCHESTER Applied Business Systems
Post Office Box 187
Briarcliff Manor, N.Y. 10510

XDMS Data Management System

The XDMS Data Management System is available in three levels. Each level includes the XDMS nucleus, VMGEN utility and System Documentation for level III. XDMS is one of the most powerful systems available for 6809 computers and may be used for a wide variety of applications. XDMS users are registered in our database to permit distribution of product announcements and validation of user upgrades and maintenance requests.

XDMS Level I

XDMS Level I consists of DEFINE, UPDATE and REPORT facilities. This level is intended as an "entry level" system and permits entry and reporting of data on a "tabular" basis. The REPORT facility supports record and field selection, field merge, sorting, line calculations, column totals and report titling. Control is via a English-like language which is upward compatible with level II. XDMS Level I $129.95

XDMS Level II

Level II adds to Level I the powerful GENERATE facility. This facility can be thought of as a general file processor which can produce reports, forms and form letters as well as file output which may be re-input to the facility. GENERATE may be used in complex processing applications and is controlled by a English-like command language which encompasses that used by Level I. XDMS Level II $199.95

XDMS Level III

Level III includes all of level II plus a set of useful DMS Utilities. These utilities are designed to aid in the development and maintenance of user applications and permit modification of XDMS system parameters, input and output of XDMS files, display and modification of file format, graphic display of numerical data and other functions. Level III is intended for advanced XDMS users. XDMS Level III $269.95
XDMS System Documentation only $10, credit toward purchase. . . $ 20.95

XACC Accounting System

The XACC General Accounting System is designed for small business environments of up to 10,000 accounts and inventory items. The system integrates accounting functions and inventory plus the General Ledger, accounts receivable and payable functions normally sold separately in other systems. Features user defined accounts, products for services, transactions, invoicing, etc. Easily configured to most environments. XACC General Accounting System (Requires XDMS, prof. Lv. III). . $299.95
XACC System Documentation only $10, credit toward purchase. . . $ 24.95

WEBTCHESTER Applied Business Systems
Post Office Box 187, Briarcliff Manor, N.Y. 10510

All software is written in macro/assembler and runs under 6809 FLEX O/S. Terms: Check, Money Order, Visa or Mastercharge. Shipment first class. Add P&H $2.50 ($7.50 Foreign). NY Res add sales tax. Specify 5" or 8".

Sales: S. E. MEDIA, 1-800-338-6800, Consultation: 914-941-3552 (evens).

FLEX is a trademark of Technical Systems Consultants, Inc.

GIMIX HAS THE 6809 SYSTEM TO SUIT YOUR NEEDS

HARDWARE

All systems feature the **GIMIX CLASSY CHASSIS**; with a ferro-resonant constant voltage power supply, gold plated bus connectors, and plenty of capacity for future expansion.

Static **RAM** and double-density **DMA** floppy disk controllers are used exclusively in all systems.

All systems are guaranteed for 2 MHz operation and include complete hardware and software documentation, necessary cables, filler plates, etc.

Systems are assembled using burned-in and tested boards, and all disk drives are tested and aligned by **GIMIX**.

You can add additional components to any system when ordering, or expand it in the future by adding **RAM, I/O**, etc.

GIMIX lets you choose from a wide variety of options to customize your system to your needs.

SOFTWARE

All **OS-9/FLEX** systems allow you to software select either operating system.

Also included is the **GMXBUG** monitor and, in systems with 128K or more of RAM, **GMX-VDISK** for **FLEX**.

All **GIMIX OS-9** systems include **Microware's Editor, Assembler, Debugger, Basic09,** and **Runb**; and the GMX versions of **RMS** and **OO** for **OS-9**.

All **GIMIX** versions of **OS-9** can read and write RS color computer format OS-9 disks, as well as the Microware/GIMIX standard format.

New and exclusive with **OS-9 GMX III** systems is the **GMX OS-9 Support ROM**, a monitor for **OS-9** that includes memory diagnostics and allows the system to boot directly from either hard disk or floppy.

A wide variety of languages and other software is available for use with either **OS-9** or **FLEX**.

OS-9 GMX III/FLEX SYSTEMS (#79)

The **#79** super system now includes (in addition to the above): the **GMX 6809 CPU III, a 256K CMOS Static RAM Board (#72), and a 3-port Intelligent Serial I/O Processor (#11).**

The **GMX 6809 CPU III** can perform high-speed DMA transfers from memory to memory and uses memory attributes and illegal instruction trapping to protect the system and users from program crashes. If a user program crashes, only that user is affected; other users are unaware of the problem.

The 3-Port Intelligent Serial I/O Board (#11) significantly reduces system overhead by handling routine I/O functions, freeing the host CPU for running user programs. This improves overall system performance and allows user terminals to be run at up to 19.2K baud.

with dual 40 track DSDD drives	**$5998.79**
with dual 80 track DSDD drives	**$6198.79**
with **#88** dual 8" DSDD drive system	**$7698.79**
with **#90** 19MB Winchester subsystem and one 80 track	**$8898.79**
with a 47MB Winchester subsystem and one 80 track	**$10,898.79**
with a 47MB plus a 6MB removable pack Winchester subsystem and one 80 track drive	**$12,398.79**

TO ORDER BY MAIL: SEND CHECK OR MONEY ORDER OR USE YOUR VISA OR MASTER CHARGE. Please allow 3 weeks for personal checks to clear. U.S. orders add $5 handling if order is under $200.00. Foreign orders add $10 handling if order is under $200.00. Foreign orders over $200.00 will be shipped via Emery Air Freight COLLECT, and we will charge no handling. All orders must be prepaid in U.S. funds. Please note that foreign checks have been taking about 8 weeks for collection so we would advise wiring money, or checks drawn on a bank account in the U.S. Our bank is the Continental Illinois National Bank of Chicago, 231 S. LaSalle Street, Chicago, IL 60693, account #73-32033.

BASIC-09 and OS-9 are trademarks of Microware Systems Corp. and MOTOROLA, Inc. FLEX and UniFLEX are trademarks of Technical Systems Consultants, Inc. GIMIX, GHOST, GMX, CLASSY CHASSIS, are trademarks of GIMIX, Inc.

OS-9 GMX I / FLEX SYSTEMS #49

The **#49** systems include 64KB static RAM, #05 CPU, #43 2 port serial board.

with dual 40 track DSDD drives	**$3998.49**
with dual 80 track DSDD drives	**$4198.49**
with **#88** dual 8" DSDD drive system	**$5698.49**
with **#90** 19MB Winchester subsystem and one 80 track	**$6898.49**

OS-9 GMX II / FLEX SYSTEMS #39

The **#39** systems include 128KB static RAM, #05 CPU, #43 2 port serial board.

with dual 40 track DSDD drives	**$4498.39**
with dual 80 track DSDD drives	**$4698.39**
with **#88** dual 8" DSDD drive system	**$6198.39**
with **#90** 19MB Winchester subsystem and one 80 track	**$7398.39**

GIMIX DOES NOT GUARANTEE PERFORMANCE OF ANY GIMIX SYSTEMS, BOARDS OR SOFTWARE WHEN USED WITH OTHER MANUFACTURERS PRODUCT.

EXPORT MODELS: ADD $30 FOR 50Hz. POWER SUPPLIES.

GIMIX, Inc. reserves the right to change pricing, terms, and products specifications at any time without further notice.

ALL PRICES ARE F.O.B. CHICAGO

Contact GIMIX for price and availability of UniFLEX and UniFLEX GMXIII Systems.

NOTE on all drive systems: Dual 40 track drives have about 700KB of formatted capacity; dual 80's about 1,400KB; dual 8" about 2,000KB. The formatted capacity of hard disks is about 80% of the total capacity.

Want to expand your system to a megabyte of Static RAM and 15 users?

Simply add additional memory and I/O boards. Your GIMIX system can grow with your needs. Contact us for a complete list of available boards and options.

#72 256KB CMOS STATIC RAM board with battery back up	**$1898.72**
#64 64KB CMOS STATIC RAM board with battery back up	**$528.64**
#67 64KB STATIC RAM board	**$478.67**
#11 3 port intelligent serial I/O board	**$498.11**
#43 2 port serial I/O board	**$128.43**
#42 2 port parallel I/O board	**$88.42**
#95 cable sets (1 needed per port), specify board	**$24.95**

TRADE UP YOUR CoCo!

GIMIX will allow you up to $1100.00 credit toward the purchase of any GIMIX system when you trade-in your working Color Computer, peripherals, and original software. The trade-in value is limited to 110% of the RADIO SHACK™ list price at the time your order is placed. You pay the freight. This offer is good only in the Continental U.S.; is limited to the first 100 orders; and expires on 9/30/84. Only one trade-in per customer.

GIMIX INC.
1337 WEST 37th PLACE
CHICAGO, ILLINOIS 60609
(312) 927-5510 • TWX 910-221-4055

The Original FLEX™ for Color Computers

- Upgrade to 64K
- RS to FLEX, FLEX to RS file transfer ability
- Create your own character set
- Automatic recognition of single or double density and single or doubled sided
- All features available for either single or multiple drive systems
- Settable Disk Drive Seek Rates
- Faster High Resolution Video Display with 5 different formats
- Save RS Basic from RAM to Disk
- Move RS Basic to RAM
- Load and save function on FLEX disk
- 24 Support Commands 12 with Source Text
- External Terminal Program

DATA-COMP has everything you need to make your TRS-80C Color Computer WORK for YOU! from Parts and Pieces to Full Ready To Use Systems. DATA-COMP designs, sells, services, and SUPPORTS Computer SYSTEMS, not just Software. CALL DATA-COMP TODAY to make your Computer WORK FOR YOU!

System Requirements

FLEX9 Special General Version + Editor & Assembler (which normally sell for $50.00 ea.) $250.00
P-MATE(RS) FLEX9 Conversion Rout. for the RS Disk Controller
when purchased with Special General FLEX9 Sys. $ 49.95
when purchased without the General FLEX9 Sys. $ 59.95

Color Computer with 64K RAM and EXT.BASIC $399.95

SPECIAL SYSTEM PACKAGES

64K Radio Shack COLOR COMPUTER, Radio Shack COLOR DISK CONTROLLER, a Disk Drive System, Special General Version of FLEX9, P-MATE(RS), and a Box of 10 Double Density Diskettes, a COMPLETE ready to run SYSTEM on your Color TV Set.

$1099.95

64K Radio Shack COLOR COMPUTER, Radio Shack COLOR DISK CONTROLLER, a Disk Drive System, and a Box of 10 Double Density Diskettes, a COMPLETE ready to run SYSTEM on your Color TV Set.

$799.95

DISK DRIVE PACKAGES, etc.

These Packages include the Radio Shack Disk Controller, Disk Drives with Power Supply and Cabinet, and Disk Drive Cable:

PAK #1 - 1 Single Sided, Double Density Sys.	$389.95
PAK #2 - 2 Single Sided, Double Density Sys.	$675.95
PAK #3 - 1 Double Sided, Double Density Sys.	$569.95
PAK #4 - 2 Double Sided, Double Density Sys.	$889.95
PAK #5 - 2 Qume Thinline Double Sided, Double Density Sys	$699.95
PAK #6 - 2 Tandon Thinline Single Sided, Double Density Sys	$599.95

PRINTERS

EPSON RX-80	$325.00
EPSON RX-80FT	$375.00
EPSON MX-100	$675.00
EPSON FX-100	$749.00
EPSON FX-80	$549.00
EPSON MX-70	$299.00

SERIAL BOARDS

MX-Series	$119.95
RX-FX-Series	$ 99.95
Spectrum MX-Series	$ 54.95

H J L Keyboards $ 74.95

FLOPPY DISKETTES

	MEMOREX	VERBATIM
5" Soft Sector Disks		
Single Sided Single Density	$2.40ea.	$2.75ea
Single Sided Double Density	$2.40ea	$2.75ea
Double Sided Double Density		$4.92ea
Plastic Storage Box		$2.00ea
8" Soft Sector Disks		
Single Sided Single Density		$3.75ea
Single Sided Double Density		$4.10ea
Double Sided Double Density		$4.75ea
Plastic Library Box		$5.00ea

PARTS AND PIECES

Radio Shack Disk Controller	$169.95	Single Drive Disk Cable for RS Controller	$ 19.95
1 Single Sided, Double Density Disk Drive Tandon	$229.95	Double Drive Disk Cable for RS Controller	$ 24.95
1 Double Sided, Double Density Disk Drive Qume	$319.95	Micro Tech. Prods. Inc. COVER CASE ROM Adapter	$ 74.95
1 Qume Thinline Double Sided, Double Density	$249.95	Radio Shack BASIC Version 1.1 ROM or 1.2 ROM	$ 34.95
1 Tandon 40 Track SSDD with Cabinet and Power Supply	$259.95	Radio Shack Extended Basic ROM	$ 39.95
Single Drive Cabinet with Power Supply	$ 79.95	Radio Shack Disk Basic ROM 1.1	$ 29.95
Double Drive Cabinet with Power Supply	$109.95	Screen Clean-Clears Up Video Distortion On Your Color Computer	$ 39.95
J & M Disk Controller	$144.95	Set of Eight 64K RAM Chips w Mod Instructions	$ 49.95

For Ordering Call TOLL FREE 1-800-338-6800

LAST OF PRODUCTION!!!

!! LIMITED QUANTITY !!

SWTPC 8212 & 8212(W)
Intelligent Terminals

New & Demo Models

At Discount Prices

Remaining Supply of
SWTPC 8212 CRT Terminals

EM-82 Video Terminal

Emulates the 8200 Series from SWTPC

MasterCard VISA

DATA-COMP

P.O. Box 794 *HIXSON, TN 37343*
1-615-842-4601

*FLEX is a trademark of Technical Systems Consultants
*OS9 is a trademark of Microware

Data-Comp—South East Media & 68 Micro Journal Are Divisions of CPI

68 Micro Journal
5900 Cassandra Smith Rd.
Hixson, TN 37343

Second Class Postage Paid
At Chattanooga, TN
ISSN 0194-5025